FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

IM2HoT: Interactive Machine Learning to Improve the House of Things

João Pedro Milano Silva Cardoso

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Hugo José Sereno Lopes Ferreira

Second Supervisor: Luís Filipe Pinto de Almeida Teixeira

February 28, 2018

IM2HoT: Interactive Machine Learning to Improve the House of Things

João Pedro Milano Silva Cardoso

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. André Monteiro de Oliveira Restivo, PhD
External Examiner: Prof. Pedro Manuel Pinto Ribeiro, PhD
Supervisor: Prof. Hugo José Sereno Lopes Ferreira, PhD

February 28, 2018

Abstract

Patterns emerge as we go about our daily home lives. Can we take advantage of these patterns, learn them and use them to make people's lives more comfortable?

Questions regarding a person's comfort in their own home, as well as security, healthcare, safety and energy savings drive smart home solutions development, with the purpose of making daily life easier, which can be achieved by remotely controlling devices or identifying relevant activities and increasing their automation.

Current home automation solutions mostly focus on remote control, monitoring, and preferences or scene settings set up by the user. Some of them are capable of learning user patterns but serve a single need. On the other hand, the autonomous control in solutions that allow for the integration of multiple devices is handled by the definition of rules, and cannot learn user patterns.

The purpose of this thesis is the development of a solution that uses machine learning algorithms to learn user patterns, that integrates multiple kinds of devices heterogeneously to serve different needs and autonomously controls them, allowing for an off-the-shelf solution better suited to the users' needs. Additionally, users are able to define what devices they want the system to monitor and learn as well as help the learning process by reinforcing choices made by the system.

This solution was achieved by extending the visual programming environment Node-Red with a new IM2HoT node. The use of a visual programming environment enables users with little programming experience to easily interact with the system.

The integration of this system into Node-Red will also allow for a flexible and easy setup, potentially reaching thousands of users.

Resumo

Padrões emergem à medida que vivemos as nossas vidas diárias. Podemos aproveitar-nos destes padrões, aprendê-los e usá-los para tornar a vida das pessoas mais confortáveis?

Questões em relação ao conforto de uma pessoa na sua própria casa, bem como segurança, saúde e poupança de energia guiam o desenvolvimento de soluções Smart Home, com o objectivo de tornar a vida diária mais fácil. Isto pode ser conseguido através do controlo remoto de dispositivos ou da identificação e automatização de actividades relevantes.

Soluções de domótica actuais focam-se principalmente em controlo remoto, monitorização e cenários ou preferências definidas pelo utilizador. Alguns conseguem aprender padrões de utilizadores, mas servem uma só necessidade. Pelo outro lado, controlo autónomo em soluções que permitem a integração de múltiplos dispositivos depende da definição de regras, e são incapazes de aprender padrões de utilização.

O objectivo desta tese é o desenvolvimento de uma solução que utilize algoritmos de machine learning para aprender padrões de utilização, que integre múltiplos tipos de dispositivos heterogeneamente para servir diferentes necessidades e seja capaz de os controlar autonomamente, permitindo soluções off-the-self que melhor se encaixam nas necessidades dos utilizadores. Adicionalmente, utilizadores são capazes de definir que dispositivos querem que o sistema monitorize e aprenda, bem como ajudar o processo de aprendizagem ao reforçar certas escolhas feitas pelo sistema.

Esta solução foi desenvolvida ao extender o ambiente de programação visual Node-Red com um novo nó IM2HoT. O uso de uma ambiente de programação visual pode também permitir aos utilizadores com pouca experiência de programação interagir facilmente com o sistema.

A integração deste sistema no Node-Red irá permitir também um setup fácil e flexível, potencialmente ao alcançe de milhares de utilizadores.

Acknowledgements

I would like to thank my supervisors, Hugo Sereno and Luis Teixeira for without their help and guidance, the project would not have achieved the present quality.

These acknowledgments are extended to the Faculty of Engineering of the University of Porto, for giving me the means and opportunity to realize this project.

Last but not least, to my mother, I would like to thank for the patience necessary during the development of the project.

João Cardoso

"Real stupidity beats artificial intelligence every time."

Terry Pratchett

Contents

List of Figures

List of Tables

Abbreviations

IoT	Internet of Things
IFTTT	If This Then That
RMSE	Root Mean Square Error
MQTT	Message Queue Telemetry Transport
QoS	Quality of Service
RF	Random Forest
NN	Neural Network
AR	Association Rules
P12	Phases 1 and 2
P1nR	Phase 1, no Repeated Entries
P12nR	Phases 1 and 2, no Repeated Entries
UCB	Upper Confidence Bounds

Chapter 1

Introduction

1.1 Context

The Internet of Things, also known as IoT, is a network of networks of devices that can interact and communicate with other devices, objects, infrastructures and the environment. From this paradigm results a large amount of data that can then be processed into actions and bring about the ability to command and control multiple devices [BK16].

IoT solutions provide competitive advantage over current solutions in multiple fields and applications, such as smart homes, smart cities, environmental monitoring, health-care, smart business, inventory and product management and security and surveillance [MSDC12].

Besides the competitive advantages it brings, IoT is a currently growing field, with Gartner [vdM17] predicting that by 2020 20.4 billion devices will be in use, and that IoT technology will be in 95% of electronics for new products designs [Pan17], and that hardware spending from consumers and businesses will be almost $3 trillion.

The solutions within the field of home automation are known as smart home solutions, which have the purpose of making people's lives easier. This can be achieved by remotely controlling devices or identifying relevant activities and increasing their automation [LCL16].

1.2 Motivation

As people go about out daily home lives, patterns naturally emerge. The lights they turn on and when, the temperature of their AC, the shutters they open and close, etc. Is there a way of using this patterns and learning them in order to make people's lives easier and more comfortable?

When looking for a solution that allows the automation of useful home behaviors, users have a wide array of choices. In some of these choices, like with the NEST Thermostat, users will find devices with the capacity to learn patterns, but they are not given any options that influence how the device learns and then makes use of these patterns. These devices also serve a single need, such as light or temperature.

There are also solutions that allow the integration of multiple devices, such as the Apple HomeKit, but these devices must be specifically compatible with the platform, which doesn't allow for an off-the-shelf, and ofter cheaper solution that might better serve the user's needs. These solutions also achieve automation scenarios by the definition of rules that indicate how devices operate together, rules that have to be hand-crafted by the users.

The motivation and challenge behind this thesis comes from the current lack of home automation solutions when it comes to learning patterns and using them to autonomously control multiple devices. At the same time involving the user into the learning process and providing options to better configure the system is an important step as revealed by an usability study of the NEST Thermostat [YN13], where the lack of understanding and options related to the learning process made the system harder to use. This solution would also allow users to assemble an off-the-self solution better suited their personal needs.

1.3 Aims and Goals

The purpose of this thesis is the development of a solution that uses machine learning algorithms to learn user patterns, that integrates multiple kinds of devices to serve different needs and is capable of autonomously controlling them using the patterns learned.

Additionally, users will be able to define what devices they want the system to monitor and learn, select the data to be used by time interval and help the learning process by reinforcing choices made by the system. This system will also deal with information in an heterogeneous way, making it possible to create an off-the-shelf home automation solution, better suited to the users needs.

The expected final result of the present work is in the form of a single node to be easily installed and used in Node-Red, a visual programming environment.

1.4 Expected Contributions

The following contributions resulted from the development of this thesis:

1. A solution capable of learning user patterns using machine learning algorithms and using these patterns to automate the behavior of multiple kinds of devices, in the form of a new node for the Node-Red platform;

2. An approach to handling data in an heterogeneous way in an home automation scenario, allowing for the possibility of an off-the-shelf solution;

3. An exploration of multiple machine learning algorithms applied to an home automation scenario. It might be of interest to note not only the methods that are successful, but also those that are not, and why.

1.5 Document Structure

Aside from the present chapter, the dissertation is structured as follows: chapter 2 presents a state of the art on IoT, smart home solutions and machine learning algorithms; chapter 3 presents an analysis on the state of the art solutions to better outline the objectives of the project and form the project proposal; chapter 4 presents the main design decisions of the project; chapter 5 presents the main implementation details; chapter 6 presents the validation testing of the system; and chapter 7 presents the conclusions regarding the main contributions and possible future work.

Chapter 2

State of the Art

This chapter presents a revision on the state of the art relevant to the multiple facets of the project of the project.

The state of the art can be divided in 3 main categories: IoT as concept, an overview of the current approaches and solutions in the field of home automation and control, and overview of the multiple kinds of machine learning algorithms.

2.1 Internet of Things - IoT

Smart objects are created by embedding electronics into everyday devices, and can be interconnected to form networks. This originates a shift from an Internet used to connect end-user devices to and Internet used to interconnect objects that communicate with each other and/or humans to form the Internet of Things. This marks the advancement towards smart spaces, such as smart cities, homes and grids [WB14], by means of ubiquitous computing [MF10].

The devices, or things, can be defined as entities that [MSDC12]:

- Have a physical embodiment
- Have a minimal set of communication functionalities
- Possess an unique identifier
- Are associated to at least one name and one address
- Possess basic computing abilities
- May possess means to sense physical phenomena or trigger action by interacting with the environment

IoT solutions can be applied to many areas, and provide competitive advantage over current solutions in multiple fields and applications, such as smart homes, smart cities, environmental monitoring, health-care, smart business, inventory and product management and security and surveillance [MSDC12].

As it is a young field of research, there are still many challenges ahead. A key aspect to IoT is the interconnectivity of devices, but the lack of communication standards make it so that devices from different vendors may implement different protocols, creating interoperability issues and making it harder to achieve a heterogeneous system.

Additionally, current IoT platforms do not possess sufficient security and privacy capabilities, a problem inflated by the large number of connected devices [MMST16]- There is also a lack of reference architectures for the design of these systems [CMV+10], and the research into design and architectural patterns is still very preliminary.

2.2 Smart Home Solutions

Connecting home gadgets with cloud capabilities is something that evolved from the area of DIY to genuine business items. Due to the advent and proliferation of smart technologies, enhanced functionality, connectivity and manageability in all sort of consumer devices such as phones, TVs and even refrigerators and in infrastructures such as cities and grids can be achieved by embedding information gathering and communication capabilities into those devices. Developers, service providers and energy utilities are seeking to extend the capabilities of these technologies beyond specific devices and to the home as a whole [WHHB15].

Ultimately a smart home would be an environment such that all lighting, heating, security, appliances and electronic devices can be controlled remotely via an application on a smartphone or computer. These control and additional monitoring capabilities have as their primary objective comfort, convenience and energy savings, improving the living experience [FFK+13].

According to Le at al. [LNB12] a smart home, the concept of which can be seen in fig. 2.1, should have the following characteristics:

- Automation - The ability to accommodate automatic devices or perform automatic functions;
- Multi-functionality - The ability to perform various duties or generate various outcomes;
- Adaptability - The ability to adjust or be adjusted to meet the needs of the users;
- Interactivity - The ability to interact with or allow for interaction among users;
- Efficiency - The ability to perform functions in a time-saving, cost-saving and convenient manner.

The following sections present some smart home solutions, divided into solitary devices and platforms or apps meant to integrate multiple devices.

Figure 2.1: Smart Home Concept Example

2.2.1 Devices

2.2.1.1 BeOn Lightbulbs

BeOn Lightbulbs [BeO17] is a smart home solution geared towards security. It's composed by a set of light bulbs with multiple capabilities.

Its main capability is learning the users lighting patterns in order to create a convincing 7-day lightning schedule. This schedule is then replayed when the user is away, in order to fool would be burglars by creating a natural simulation of the presence of the home's occupants.

Figure 2.2: BeOn Lightbulbs

It can also be used to replay lightning patterns when the homes doorbell is rung, turn on the lights when a smoke detector is heard, and provide up to 5 hours of backup lighting in the case of blackouts. Additionally, it provides a smart phone app to visualize the schedule and configure other settings, such as if the user is home or away and what light patters to replay.

2.2.1.2 NEST Learning Thermostat

The NEST Thermostat [NES17b] is a smart thermostat with learning capabilities and a host of features such as schedule learning, remote access, occupancy sensing, and eco-feedback. It also provides a smart phone and web-based dashboard where users can see the history of when and how long the system ran, and control various settings.

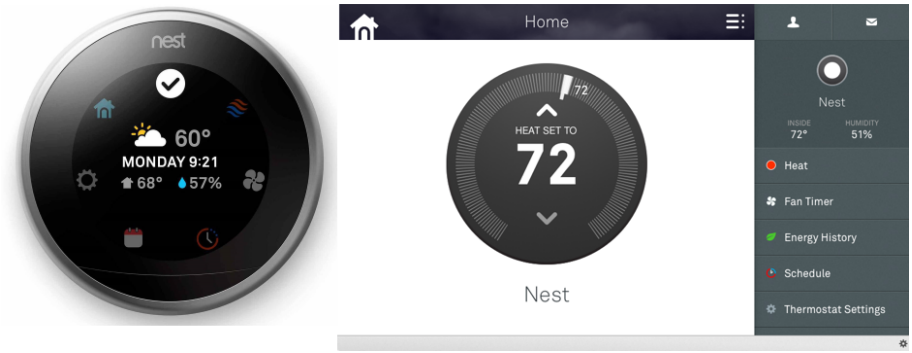

Figure 2.3: NEST Learning Thermostat and Interface

One of its features, Auto-Schedule makes use of machine learning to analyze user patterns regarding temperature changes and automatically create a temperature schedule. This algorithm takes a week to generate the initial schedule and after that keep adjusting it to adapt, which can then be reviewed by the user [YN13].

The NEST Thermostat allows integration with several other products, such as Google Home, a voice activated speaker powered by Google Assistant, Amazon Alexa, Wink, Whirlpool Refrigerators and many others [NES17a].

2.2.2 Platforms and Applications

2.2.2.1 Apple HomeKit

The Apple HomeKit [App17b] is a platform that aims to integrate multiple devices from different vendors and present them all under a single interface for smart phone or tablet, creating a custom home automation configuration. This allows users to control, configure and communicate with their devices using Siri, or the Home app.

In Apple HomeKit, a home is defined as any physical location of relevance to the user. This home can then be arranged into rooms, such as *Bedroom* or *Living Room*, which provide a simple way to organize devices and have Siri recognize them by name. Zones allow rooms to be organized into groups, the same way rooms work in regards to devices.

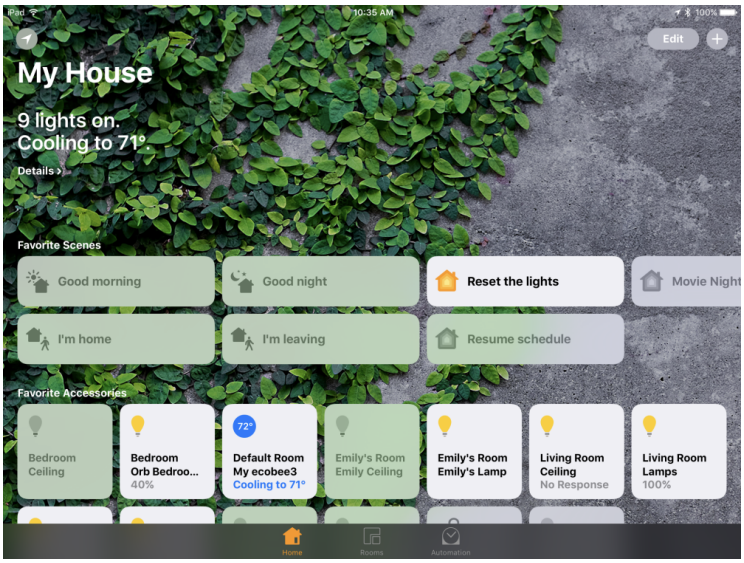

Figure 2.4: Apple HomeKit Interface Example

This system also allows the definition of Scenes, user defined scenarios where multiple devices are controlled at the same time when commanded by the user. In addition, Automations provide a way for devices to react to situations, like turning on the lights when it gets dark outside, or when the user opens the front door. However, only specific devices come enabled with HomeKit compatibility. [App17a]

2.2.2.2 Samsung SmartThings

Samsung SmartThings [Sam17] is a system similar to the Apple HomeKit. It consists of a hub that allows the interconnection of multiple devices in order for them to work together and achieve a home automation scenario.

This solution allows such scenarios by defining routines where multiple devices work together, such as turning off lights when no presence is detected in a room or turning on the radio, lights and adjusting the thermostat in the morning.

Users have a SmartThings app [Sam] available for Google Assistant, Amazon Alexa, Apple and Android devices that already comes with some predefined routines. As is the case with Apple HomeKit, only select devices are compatible with Samsung SmartThings.

2.2.2.3 MyFox

MyFox [Som] is a smarthome solution focused on security. It is composed of a camera, siren, a vibration sensing device for doors or windows and a hub to which the devices are connected.

In MyFox, automation capabilities come from an exterior service, IFTTT, which enables users to create their own rules and scenarios, such as sending a notification to the phone when motion is sensed in one of the cameras, without demanding from them programming experience.

Figure 2.5: Samsung SmartThings Starter Kit

Figure 2.6: MyFox Security Camera and Home Alarm

IFTTT (If-This-Then-That) [IFT] is a web service that, even though it does not achieve anything on it's own, connects other services to each other, allowing them to be combined such that tasks can be accomplished. IFTTT describes actions by using recipes, many of which already come pre-made only need to be configured to be used. These recipes are easy to setup, giving even users with little programming experience many options.

2.2.2.4 Wink

Wink [WIN17], a free app available for iOS and Android, is a smart home platform that aims to bring together products from different vendors, such as NEST, Phillips, Schlage and Chamberlain.

This app allows the user to see the status of each home device and remotely control them, as well as creating Shortcuts, that allow the control of multiple devices at the same time, like turning off multiple lights with a single command. It also allows the creation of Robots, automatic commands such as turning on the lights when the front door is unlocked. [WIN17]

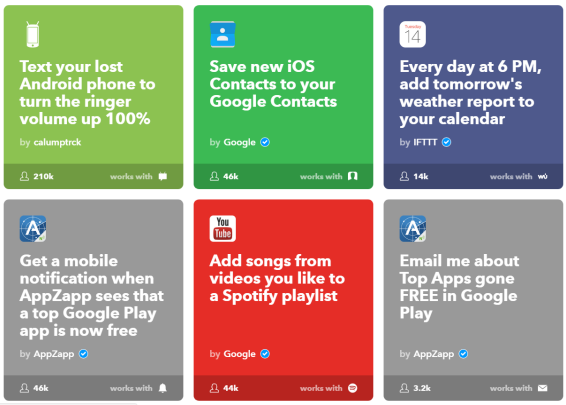

Figure 2.7: Example of IFTTT's pre-made recipes

Figure 2.8: IFTTT Displaying a Finished Rule in MyFox

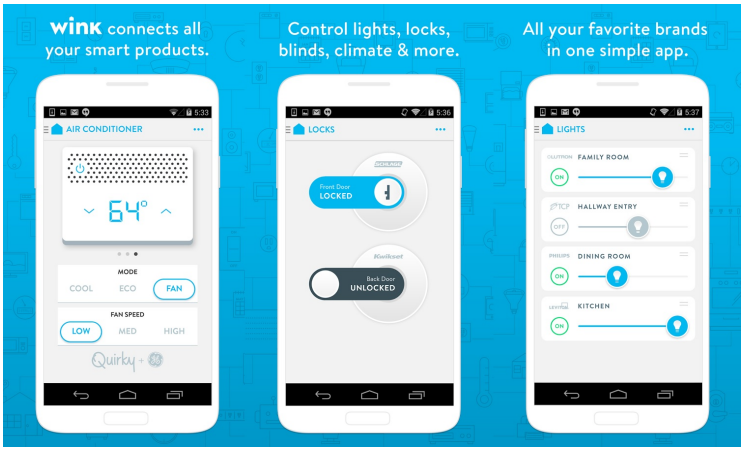

Figure 2.9: Wink App Interface

2.2.3 Other Solutions

This section focuses on more experimental solutions, that don't have products available for users in the marketplace.

2.2.3.1 An IoT System to Estimate Personal Thermal Comfort

Laftchiev and Nikovski [LN16] detail an IoT system with temperature as its focus, which uses machine learning to create a personalized model for thermal comfort, the interface of which can be seen in fig. 2.10.

Two approaches to the problem are explored. The first approach consists of using machine learning classification algorithms, some of which are detailed in section 2.3, to determine which of 7 user comfort states, based on Fanger's model of thermal comfort for a group of individuals, is appropriate. The second approach consists of using a regression function to predict a continuous value.

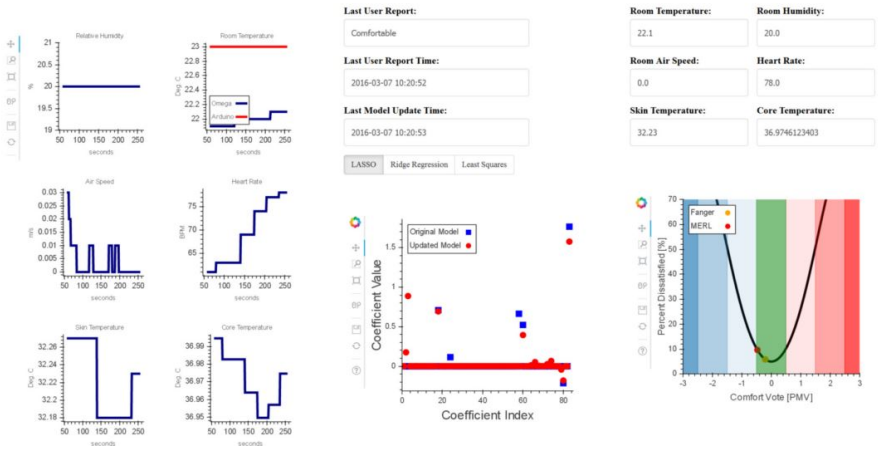

Figure 2.10: Laftchiev and Nikovski's system interface

For validation, this project made use of the leave-one-out cross validation method to find the average root mean square error, or RMSE, of the prediction, with a 20/80 split for training and cross validation. The results are presented in table 2.1, as transcribed from [LN16].

2.2.3.2 CASAS: A Smart Home Solution in a Box

The Smart Home in a Box is a project developed by the Washington State University Center for Advanced Studies in Adaptive Systems (CASAS), with the goal of creating a smart home solution that can be self-installed [HHKT16].

Its architecture was designed to be easily extended, scaled and maintained, as seen in fig. 2.11, and it can perform activity recognition by processing the information generated by the sensors by mapping a sequence of data to a corresponding activity label. The sequence of events in Fig.2.12 could, for example, be mapped to the *Sleep* activity.

The system uses a designed SVM method for real-time activity recognition, having found it performed better than naive Bayes classifiers, hidden Markov models and conditional random fields [CCTK13]. However, it does not possess any means of controlling, configuring or automating devices, merely to gather activity data.

Method	RMSE
SVM	0.560
Kernel Ridge Regression	0.574
Logistic Regression	0.575
Support Vector Regression	0.585
Bayesian Ridge Regression	0.589
Ridge Regression	0.597
LASSO	0.601
ARD	0.608
Linear Discriminant Analysis	0.621
Ordinary Least Squares	0.624
Elastic Net	0.622
KNN	0.634
Gaussian Process Regression	0.701
Least Angle Regression	0.710
Quadratic Discriminant Analysis	0.885
Fanger's Method	1.15

Table 2.1: RMSE of Thermal Comfort Estimation Methods

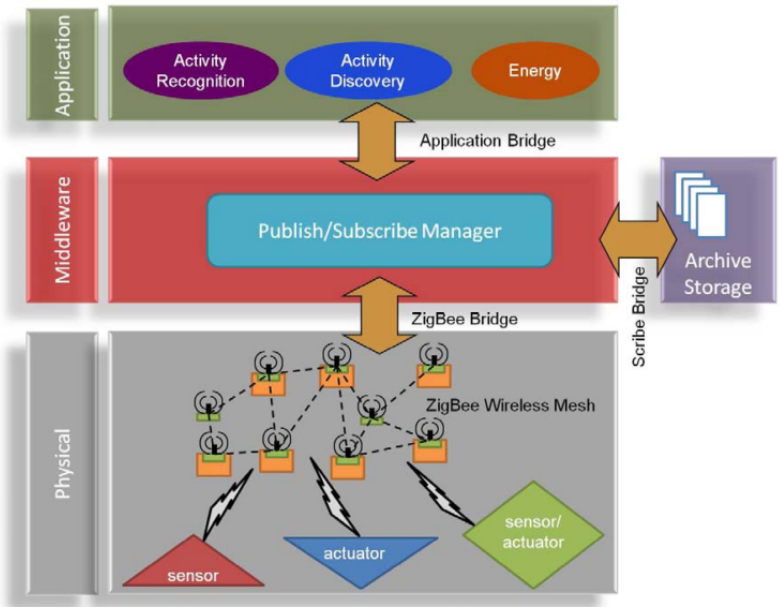

Figure 2.11: CASAS Smart Home in a Box Architecture

2011-06-15 03:38:23.271939 BedMotionSensor ON
2011-06-15 03:38:28.212060 BedMotionSensor ON
2011-06-15 03:38:29.213955 BedMotionSensor ON

Figure 2.12: CASAS Smart Home Sensor Data Example

2.3 Machine Learning

Machine Learning is a subset of artificial intelligence focused on algorithms that allow computers to learn. These algorithms are given a set of data, also knows as training set, from which they are supposed to infer information about its properties and create a model, which is possible due to patterns contained within the data. This model then allows the algorithm to make predictions about other data that it might see in the future [Seg07].

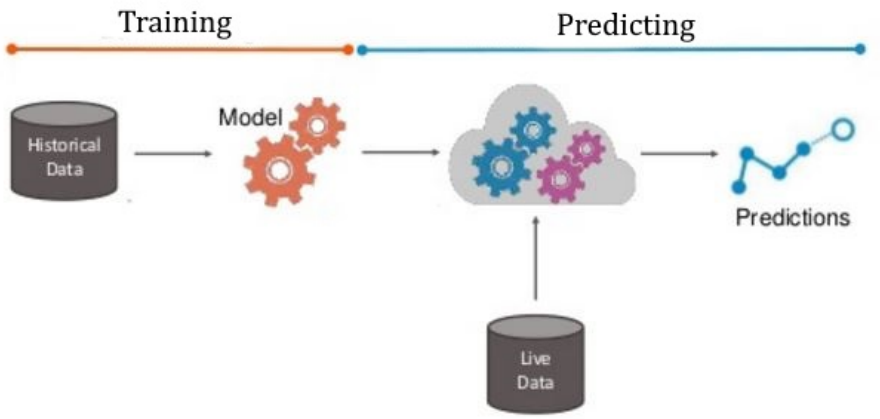

Figure 2.13: Machine Learning Training and Prediction

The problems of pattern recognition machine learning aims to solve are important in a variety of fields, such as biology, psychology, medicine, marketing, computer vision artificial intelligence and remote sensing [JDJ00].

For all of their advantages, machine learning methods are not without fault. Being deployed in constantly changing and evolving environments means the models must be constantly updated. An additional challenge in their large-scale use is that the models cannot be shared between users or reused in contexts different than the setting in which the training data was collected [KGV16].

Something to watch out for when dealing with machine learning is the phenomenon of overfitting. An algorithm creates a model based on training data, but if the model adapts too well to the training data instead of finding a general predictive rule, it might becomes incapable of correctly classifying new cases [Die95].

The following sections address multiple types of machine learning and their relevance for the project.

2.3.1 Supervised Learning

In supervised learning, the training set is composed of a set of inputs and expected outputs, and an algorithm is used to discover the mapping between them. When using one of these methods, a set of inputs is entered into the model, which then predicts the state of the outputs based on what it has learned [Seg07].

The inputs themselves are composed of a set of features, the characteristics deemed relevant for the classification of an object, such as its shape or color. For each of these inputs there is an output, which is the correct classification or value. Supervised learning problems can be divided into two groups: regression, when the expected output is quantitative and classification, when the expected output is qualitative.

2.3.1.1 Decision Tree

Decision Tree, an example of which can be seen in fig. 2.14, is one of the simplest machine-learning methods, and can be used for both regression and classification.

It breaks the training set down into incrementally smaller subsets, while at the same time developing the decision tree. This results in a series of if-then statements arranged as a tree composed of decision nodes and leaf nodes. A decision node has two or more branches and a leaf node represents a classification or decision.

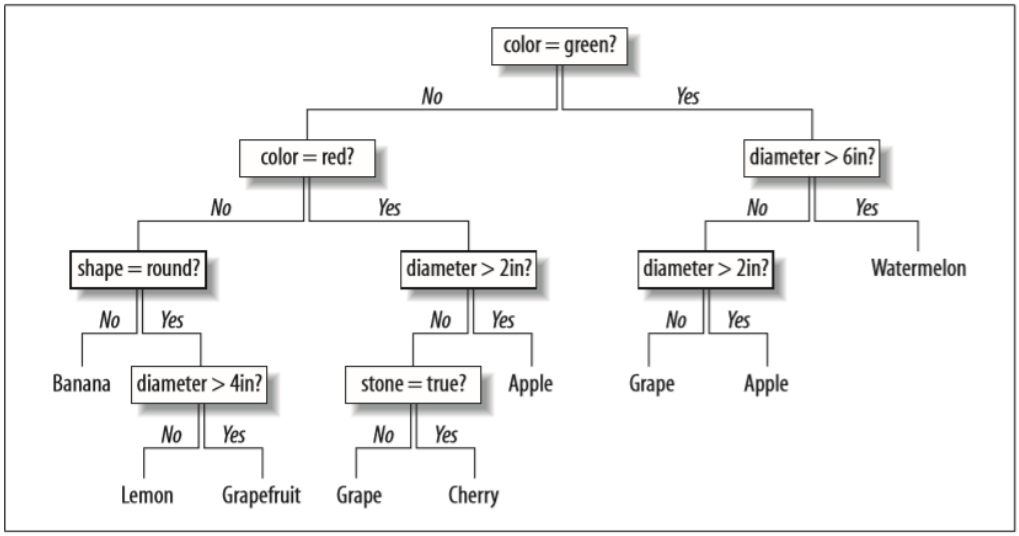

Figure 2.14: Example Decision Tree

When compared to other machine learning algorithms, decision trees are easy to interpret visually, which can help in other planning endeavors outside of the classification effort. It can also bring important features to the top of the tree, making it useful not only for classification, but also interpretation. It also allows the combination of categorical and numerical data, which can be useful for many classes of problems [Seg07]. This aspect is of particular relevance for the project, as seen in section 4.8.

However, they are not as good when it comes to making predictions for numerical outputs. Additionally, decision trees do not support incremental training, having to start from the beginning when new elements are introduced to the training set, which can result in a very different tree. They can also become extremely large and complex, which hinders the classification process, potentially making it slow. [Seg07]

2.3.1.2 Random Forest

Random Forest is a method that can be used for both regression and classification, by making use of an ensemble of decision trees. Ensemble learning results from the weighting and combination of multiple individual models in order to improve on the stability and predictive power of the system. This combination of individual models contributes to the overall accuracy and gives better results than when using the models individually [ACWR16], which is why ensemble was also used, as described in section 4.9.4.

In this method, an example of which can be seen in fig. 2.15, each tree gives a classification for the same input data, the end result being the result with most votes. [LB04]

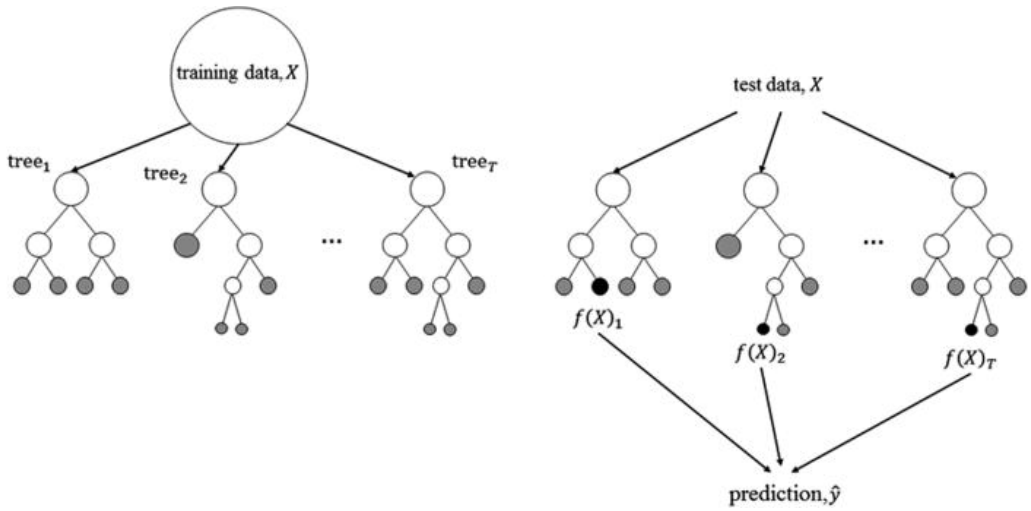

Figure 2.15: Example Random Forest

The random forest algorithm offers great accuracy, is efficient on large data sets, can handle thousands of input variables and does not overfit data [LB04]. This algorithm will be used for the training and prediction aspects of the project, as described in section 4.9.1.

2.3.1.3 K-Nearest Neighbor

K-Nearest Neighbor, also known as kNN, is a method that can be used for both regression and classification, and one of the most widely used, since it's simple and easy to implement [JDJ00].

For the classification of a testing sample, kNN calculates the distance between the testing sample and all training examples, in order to identify the K closest neighbors, as seen in fig. 2.16. There are multiple distances available for this step, out of which Euclidean is the most widely used [HHKT16]. After the k closest neighbors are established, the testing sample receives the classification via voting, taking into account the classification of its neighbors.

One of the biggest advantages of kNN is that new training data can be added at any time without retraining, unlike other algorithms. However, this leads into its major weakness. Since all

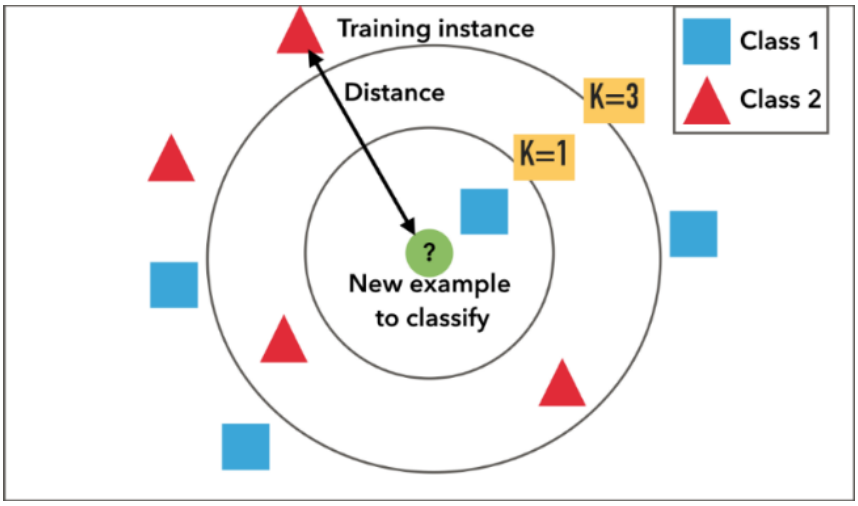

Figure 2.16: KNN Algorithm Example

of the training data is required to make predictions and every item a prediction is being made for is compared to all others in the training data, both time and space issues might arise [Seg07].

2.3.1.4 Naive Bayes

The Naive Bayes algorithm is based on Bayes' probability theorem. which describes the probability of an event based on knowledge of previous events, as shown in equation 2.1.

$$P(A \mid B) = \frac{P(B \mid A)\,P(A)}{P(B)} \tag{2.1}$$

Where A and B are events, P(B) is not zero, P(A|B) is the probability of event A occurring given that event B is true and P(B|A) is the probability of event B occurring given that event A is true. The naive aspect of the algorithm is the assumption that the features are mutually independent. This often isn't true, but even so this algorithm has been shown to work well for some classes that have a high degree of feature dependency [Ris06].

This algorithm can be trained and queried quickly. It is also simple to interpret what the classifier has learned, and see what features are best for classification by looking at the probabilities, which can also be used for other applications. However, by assuming that features are independent, it cannot learn how they interact with each other [Seg07].

Even though it was not used in the project, it might be a method worth considering since the final features that compose the training data, as described in section 4.4, are independent from each other.

2.3.1.5 Support Vector Machine

Support Vector Machine, also known as SVM, is a method used for classification problems. With this algorithm, the model is built by finding the dividing line between two categories, as seen in

fig. 2.17. While there are multiple lines that could fit this criteria, SVM chooses the one that maximizes the margin between the two instances closest to the line. This line is then the only necessary aspect to classify new instances [Seg07].

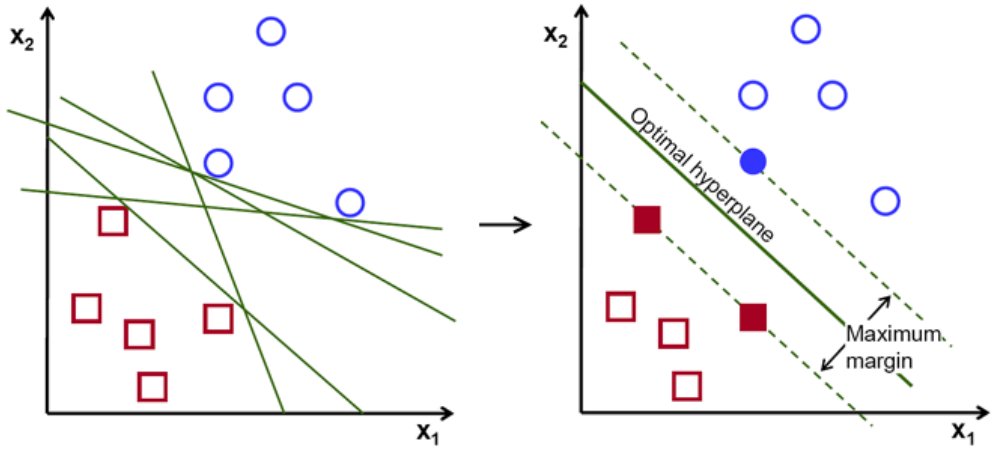

Figure 2.17: SVM Hyperplane Example

There are situations, however, where the training data cannot be linearly divided, in which case SVM uses what is known as the kernel trick. With this mapping function, it's possible to map the features to a high-dimensional feature space where it's easier to find a hyperplane that divides the categories.

SVM originally separates two classes, but there are many problems that require more categories. In order to be able to distinguish between more than two classes, the SVM algorithm can take one of two approaches: one-vs-one or one-vs-all. Both of these approaches decompose the problem into a series of binary problems, so that SVM can be directly applied [WX14].

2.3.2 Artificial Neural Networks

The goal of a neural network is to approximate a function f. Where a classifier would map and input x to a category y via a y = f(x) function, a neural network would define the mapping with y = f(x;θ), where θ is the parameter the network learns to get the best approximation [GBC16].

These networks are composed of multiple functions organized in layers, and can be represented by an acyclic graph that shows how the functions are connected, where the first layer is the input layer and the final is the output layer. The remaining layers are called hidden layers.

Fig. 2.18 shows a mostly complete chart of neural network topologies [Vee16].

The IM2HoT project will use a feed forward neural network. In these networks, information flows from the input to output layers without any feedback connections in which the outputs are fed back into itself. This algorithm will be used in the training and prediction aspects of the project, as described in section 4.9.2.

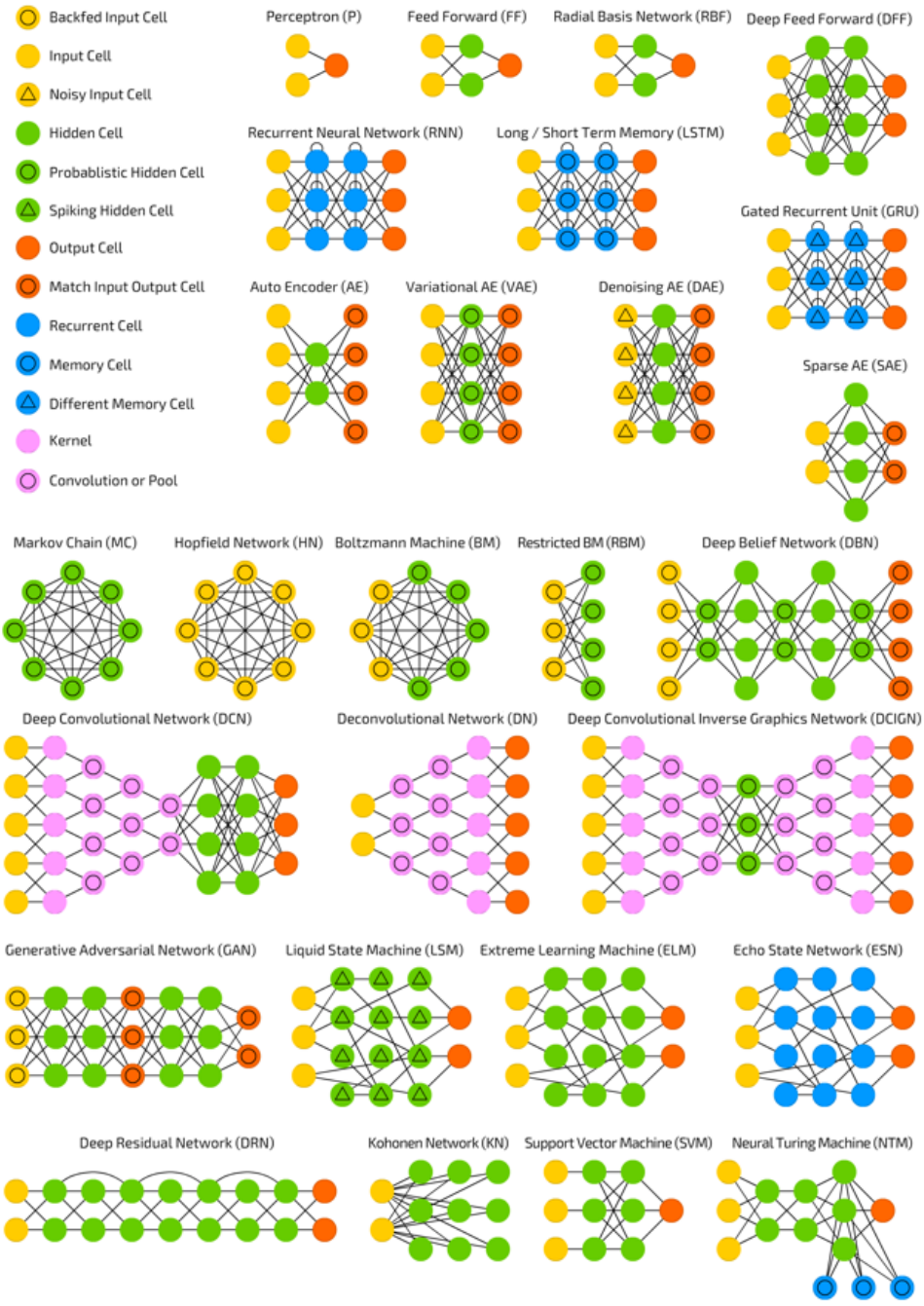

Figure 2.18: A Mostly Complete Chart of Neural Networks

2.3.3 Unsupervised Learning

Unlike supervised learning, unsupervised learning methods are not trained with examples of correct answers, not having labeled data. Their goal is instead to find structures within the data, and are not concerned with classification.

These machine learning methods are not very useful considering the context of the project.

Considering one of the objectives is to have the system control devices by making predictions based on the users previous behavior, supervised learning is more apt for this project.

2.3.4 Reinforcement Learning

Reinforcement learning is another category of machine learning, one that shares some commonalities with supervised learning. It's still the mapping of inputs to outputs, but the expected output is not known.

Instead, the algorithm aims to take actions such that it achieves the greatest reward, and must discover what actions to take and the rewards associated by trying them [Sut92]. As such, an agent that makes use of reinforcement learning must explore the environment in a trial and error basis, in the context of the problem, while an agent using supervised learning does not. From this exploration results a policy, which maps states to actions. The optimal policy is the one that yields the greatest rewards over all possible states.

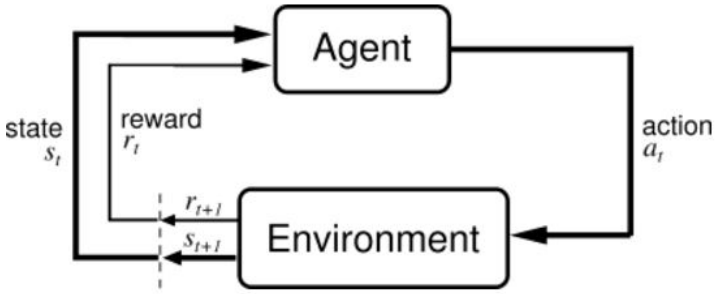

Figure 2.19: Simple Reinforcement Learning Cycle

Reinforcement learning can provide a more flexible approach than supervised learning in some cases. It can be applied in situations where domain knowledge, which is necessary to give supervised learning algorithms the correct answers and form the training dataset, is unavailable or incomplete [GE11].

2.3.4.1 Q-Learning

Q-learning is a reinforcement learning algorithm that can be used to solve Markov Decision Processes [WD92], which are a class of sequential decision processes where the cost and transition functions depend only on the current state of the system and the current action. [Put90]

The Q-Learning algorithm is defined by equation 2.2.

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha[r_{t+1} + \lambda maxQ(s_{t+1},a) - Q(s_t,a_t)] \tag{2.2}$$

Where $Q(s_t,a_t)$ is the Q function to be updated, based on state s and action a at time t, α is the learning rate, a number from 0 to 1 that signifies the extent to which new information override old information, r_{t+1} is the reward earned when transitioning to the next turn, λ is the discount rate, an

number from 0 to 1 which determines how much future rewards are worth when compared to the value of immediate rewards and $maxQ(s_{t+1}, a)$ is the value of the action that is estimated to return the largest total future reward, based on all possible actions that can be done. This algorithm is very simple to implement and does not require a model.

In Q-learning, the agent's routine consists of a sequence of distinct stages [WD92]:

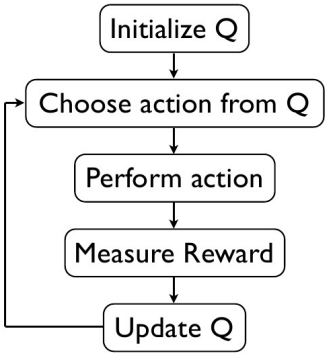

Figure 2.20: Q-Learning Cycle

2.3.4.2 Multi-Armed Bandit

Multi-armed bandit problems have been used to model the trade-off of an automatic agent which wants to obtain knowledge by exploring the environment, but also exploit it using the reliable knowledge it already has [KP14], and can be use for reinforcement.

The problem is based on gambling using slot machines, also known as one-armed bandits. A gambler has at his disposal k slot-machines, and his objective is to maximize his reward. Each time he selects a machine he receives one. Since each arm has a different distribution of rewards, the gambler wants to find the arm that gives him the best expected reward, and keep using it.

Multi-armed bandit is an example of exploitation vs exploration. If the user plays exclusively on the machine that has given him the highest reward so far (exploitation), he may not be able to discover if another machine ends up having a better return. However, if the spends too much time trying out all the machines (exploration) he may not be able to play the best arm as often as is desirable.

2.3.4.2.1 Upper Confidence Bound

The UCB algorithms were introduced as a simpler and more elegant solution for multi-armed bandit problems, implementing the idea of optimism in the face of uncertainty [KP14].

The simplest of these, UCB1, saves the number of times each arm has been played in addition to the reward mean. Each arm has to be played once to initialize the rewards. After that, in round t, the algorithm greedily picks the arm using the formula:

$$arm(t) = \max_{i=1...k} \left(\mu_i + \sqrt{\frac{2lnt}{n_i}} \right) \tag{2.3}$$

Where μ_i is the mean of the rewards of arm i and n_i is the number of times the arm has been chosen so far. This method will be used for the reinforcement aspect of the project, as described in section 4.7.1.

2.3.5 Association Rules

Association rules learning is a data mining method for retrieving relations between variables in large datasets. This method was first introduced for discovering relationships between products taking into account transactions recorded in a supermarket's database in order to inform decisions such as sales, coupons and where to place merchandise in order to increase profit [AIS93].

Formally, a set of binary attributes called items are defined. Each transaction in the database is represented as a binary vector, 1 indicating the item is present and 0 otherwise.

An association rule is the implication in the form X => Y, where Y is an item not present in X. There is an interest lies having rules follow certain constraints [AIS93]:

- Syntactic Constraints: These constraints involve restrictions on items that can appear in a rule. There might only be interest in rules that have a specific item appearing in the consequent or antecedent.

- Support Constraints: These constraints concern the number of transactions in a database T that support a rule. The support for a rule is the fraction of transactions that have the same antecedent and consequent. A motivation for support constraints comes from the fact that there is only usually interest in rules with support above some minimum threshold.

This method will be used in the training and prediction aspects of the project, as described in section 4.9.3. Since it does not matter the type of value being used for the variables, it should be able to deal with numeric and string values, which is an advantage considering the logic described in section 4.8. These rules are also readable by humans, and can be saved for later use.

2.4 Conclusions

Current smart home solutions focus more on remote control, monitoring and automation scenarios defined by the user. In the cases where the integration of multiple devices for multiple needs is a possibility, they need to be from specific vendors in order to be compatible, which doesn't allow for a more personalized, off-the-shelf solution that might better suit the users needs.

It is possible to identify a lack of solutions that have the capacity to learn patterns and are flexible enough to handle multiple kinds of needs.

Machine learning algorithms have multiple uses within the IoT field, such as analyzing past sales data to predict customer behavior, extract knowledge from bioinformatics data and optimize robots so that they complete a task using minimum resources, making them suitable candidates to use in this project.

Chapter 3

Thesis Statement

This chapter presents an analysis of the smart home solutions presented in chapter 2 in order to detail the main expected contributions of the IM2HoT project.

3.1 Main Contributions

Solutions such as Apple HomeKit and Samsung SmartThings provide users with remote access and monitoring capabilities towards their devices, and allow the user to design automation scenarios by specifying how devices should interact, alongside the relevant triggers, but present no capacity when it comes to learning and users patterns. These solutions allow the integration of specific devices, enabled to be compatible with the platform, that serve multiple needs, such as temperature and light. Other solutions, like the NEST Thermostat and the BeOn Lightbulbs have the capacity to learn the users patterns, but serve a single need.

Current solutions are mostly focused on remote control and monitoring capabilities, and whatever automation exists is handled by the definition of rules or scenarios setup by the user. In the case where the integration of multiple devices for multiple needs is a possibility, they need to be from specific vendors in order to be compatible, which doesn't allow for a more personalized, off-the-shelf solution that might better suit the users needs.

It is possible to identify the lack of a solution that learns users patterns and then uses them to achieve device automation, while at the same time allowing the integration of multiple devices in a homogeneous way.

As such, the main contributions of the IM2HoT project are as follows:

1. Learn user patterns - The system is able to learn simple user patterns by using machine learning algorithms;

2. Automate devices using patterns learned - The system is able to take control over devices by using the models created;

3. Allow the user to participate in the learning process - With a reinforcement mechanic, the user can mark system decisions as incorrect or incorrect;

4. Deal with devices heterogeneously - The project with employ a logic that treats devices in an homogeneous fashion.

The project makes use of Node-Red, a visual programming environment, to develop the system. In Node-Red, the solution takes the form of a single, easily installable node, which the user can drag, drop, connect and configure to develop their application.

3.2 Validation Methodology

The project does not be make use of any available datasets. Instead, simulated data was generated for the creation of the machine learning models and validation. This synthetic dataset simulates simple activities of a person over the course of one week in a specific scenario by using a custom made script and imported into the InfluxDB database.

By not using an already available dataset, there is no need to interpret or format it to be used, it can be created to better serve the project in terms of what inputs and outputs should be present and it becomes easier to see the connections between devices and what the expected output should be. Considering that the aim of the project is the creation of a system that aims to make predictions, multiple algorithms will be compared, and their accuracy measured by running tests made for each scenario.

Chapter 4

Design

This chapter will address all relevant design decisions, starting with a decomposition of the system based on previously delineated objectives and requirements.

4.1 System Decomposition

Taking into account the objectives described in section 3.1, the system can be divided into the following facets:

- User Options - Several options related to device selection and learning algorithms;
- Database Access - Information is retrieved from a database and processed to extract the features and organize the training data;
- Training Data Creation - Training data is created based on the retrieved information and the algorithms are trained, which deals with objectives 1 and 4;
- Prediction - Input device information is processed to extract the features and create the prediction data, which is given to the algorithms to obtain a prediction, which deals with objective 2;
- Reinforcement - Provide methods for user reinforcement, which deals with objective 3.

The collaboration diagram of the entire system can be seen in Figure 4.1.

When the user interacts with a device, those actions are sent to the IM2HoT system, and saved in the InfluxDB database. Having this information updated is important because of the training and predictions aspects. Some information, namely the rules created by the Association Rules method and the reinforcement reward information are saved in files instead.

Upon being instructed to train, the system retrieves the necessary information, creates the training data and trains the algorithms. If an action arrives after an algorithm is trained, it will trigger a prediction, which might be sent directly to the output device or the user as a Slack app notification. If notified, the user can accept or reject the decision, and the rewards are updated.

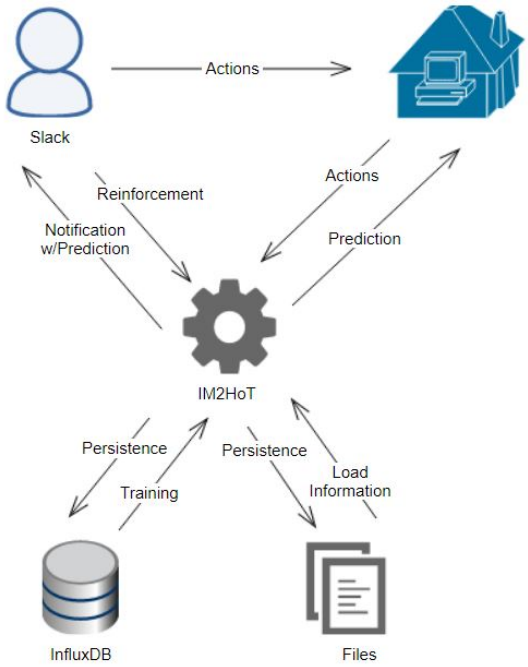

Figure 4.1: IM2HoT Collboration Diagram

4.2 User Options

User configuration is paramount to the operation of the system. Not only are there relevant options that need to be adjusted, it will also allow the system to make use of apriori knowledge that the user can provide.

An example of this knowledge is the definition of the input and output devices. The system has to learn how those devices are related, but the user selected them for a reason. This allows not only to narrow down relations, but also avoid undesirable relationships that might occur between devices by pure coincidence.

At the start of the project, the following options where identified as necessary:

- Database Configuration - Options related to database connection;
- Start and Ending dates - Time frame for data retrieval in order to train the algorithms. This way the user can freely select any time interval, defined by year, month, day, hour, minute and second;
- Input Devices - A list of devices whose data will decide the behavior of the target device;
- Output Device - The target device, to be selected by the user;
- Automatic Mode - Allows the user to decide when the system can make decisions upon receiving data.

Additional options will be expanded upon as the development is described, and summarized in section 4.11.

4.3 Database

The device data is kept in an InfluxDB database exterior to the system. Each series must have the following format:

```
time | device_id | type | value
```

Where *time* is the timestamp of when the input was recorded, *device_id* is the unique identifier of the device, *type* is the kind of device which must match the series and *value* is the recorded value of the device at the time.

The data is first retrieved for all devices in a user defined time interval, and is then processed to create the training data.

4.4 Training Data

As mentioned in section 2.3.1, training data for supervised learning is composed of pairs of inputs and outputs, where the inputs are the relevant features extracted from data and the output is the desired result for that set of features.

Feature extraction is therefore the first step to creating the training data. As for the output, what the algorithms aim to learn is the state of the output device selected by the user.

4.4.1 Starting Case - Binary Values

The design of the training data started with a simple example: a person sits down and turns on a light when it's dark. This example makes use of a presence sensor as input and a light sensor as output.

The starting principle is that the behavior of the target device, which consists of transitions from on to off or the opposite, happen as a consequence of the behavior of the input devices before the transition. Fig. 4.2 exemplifies a possible behavior of the user. The presence sensor is activated for a short time first, and then for a longer interval as well as the light.

The closest event in time and before the events of the target device is responsible for said transition. Action 3 is related to event A and action 4 is related to event B. Events 1 and 2 have no effect on the output device and are ignored. In order to avoid having to compare the output device entries to all other entries in order to find the closest one, queries retrieve data ordered by time, in a descending fashion. From this results the following data:

```
presence: 1 | light: 1
presence: 0 | light: 0
```

This give a direct relation between the devices that indicates that the light is turned on when the sensor detects something and is turned off otherwise.

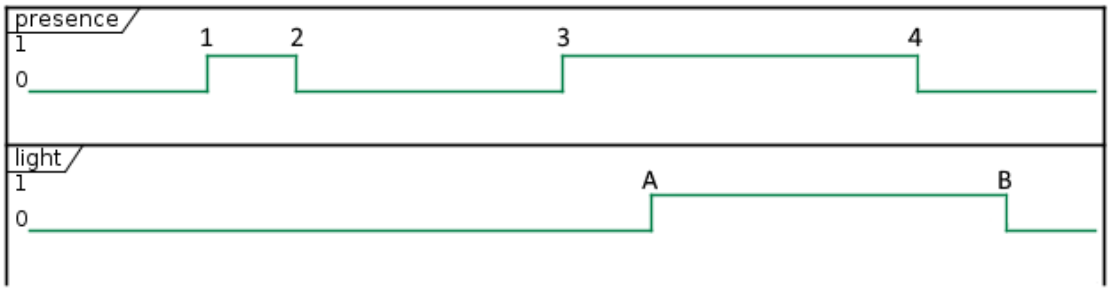

Figure 4.2: User Behavior Example for Simple Scenario

4.4.2 Time Sensitivity

What would happen if the presence sensor detected a presence an hour before the light was turned on? Does it make sense to say the light was turned on as a consequence of an event that happened that long ago?

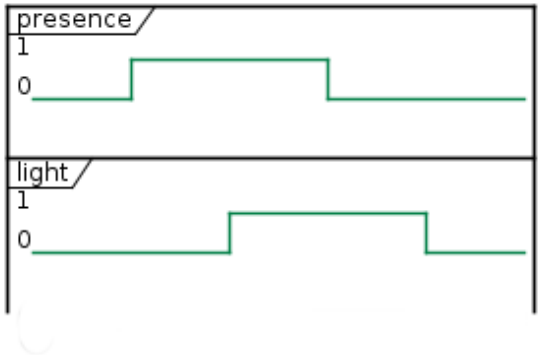

Figure 4.3: Starting Case - Different Behavior

The situation is different, but the resulting training data would be the same as in Figure 4.2.

To solve this issue, a causality window with a duration of seconds, to be defined by the user and centered on events of the target device, was introduced so that the system can better determine how data from the input devices is related to the transition, as seen in fig. 4.4.

A transition feature related to each input device was implemented. The transition is considered active if inside the causality window, or inactive otherwise, which results in the data:

```
From event A:
presence: 1 | presence_t:0 | light: 1
From event B:
presence: 0 | presence_t:0 | light: 0
```

By contrast the data of the behavior of fig. 4.2 would be:

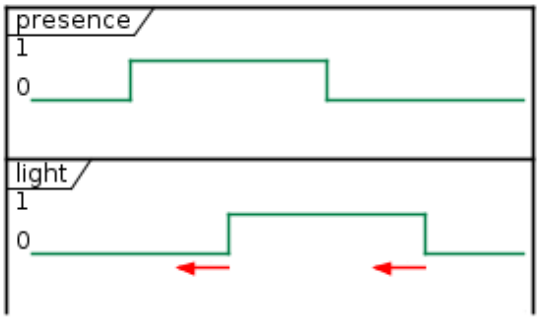

Figure 4.4: Starting Case - Different Behavior with Causality Windows

```
presence: 1 | presence_t:1 | light: 1
presence: 0 | presence_t:1 | light: 0
```

Allowing for the distinction to be made. In the event no data is found inside the causality window, the most recent value is used instead, and the transition is considered inactive.

4.4.3 Unused Information

Currently, only the information regarding when the target device is turned on or off is being used, with the remaining information being ignored. However, supervised algorithms such as random forests and neural networks and will always try and give an answer, so the more useful information it has, the better. This means that it might be important to not just give it data about the scenarios in which the device is turned on or off, but also when it is supposed to continue being on or off as the inputs change, which is to say, when it's supposed to keep the previous state and do nothing.

To this end, a second phase when processing the database results is introduced. Here, it's taken into account what happens in the entries of the input devices that are not inside any causality window.

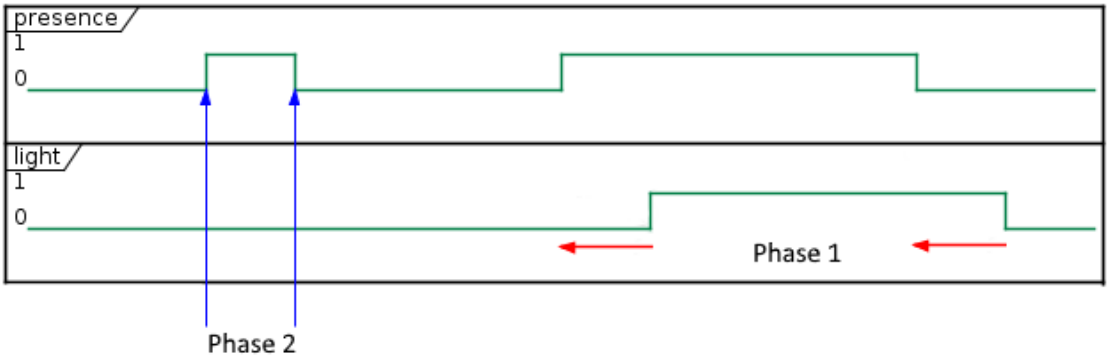

Figure 4.5: Phase 1 and Phase 2

```
Phase1:
presence: 1 | presence_t:1 | light: 1
presence: 0 | presence_t:1 | light: 0
Phase 2:
presence: 1 | presence_t:1 | light: 0
presence: 0 | presence_t:1 | light: 0
```

In this phase, all entries not processed in phase 1 are used as anchor, and the most recent entries of the other input devices in relation to the anchor and the anchor itself are used to create further training data.

The present examples of phase 2 assume that there is always at least one entry in the data that gives the most recent value of the other devices. However in reality, when going further back into retrieved results, at a certain point there might not be any data. As such, if not enough information is found for all the features, no data is created.

In the example of fig. 4.5, in the first instance of phase 2 the value of *presence* is 1, and the most recent entry of *light* gives a value of 0. In the second instance the value of *presence* is 0, and the most recent entry of *light* again gives a value of 0.

There is however a clear contradiction in the data, between the first lines of each phase. Did the user sit down and decide to stand up again, so he didn't turn on the light? Did the user sit down, but didn't need to turn on the light because he could see without it? Was it the time of day? Was some other light turned on? In order to achieve more detailed behavior more information is required, in the form of useful input devices to the situation at hand.

4.4.4 Maintaining the Previous State

In the example of fig. 4.6, the light is turned on, and multiple events occur that have no effect on the output. The light is then turned off, and the same events occur, once again having no effect on the output.

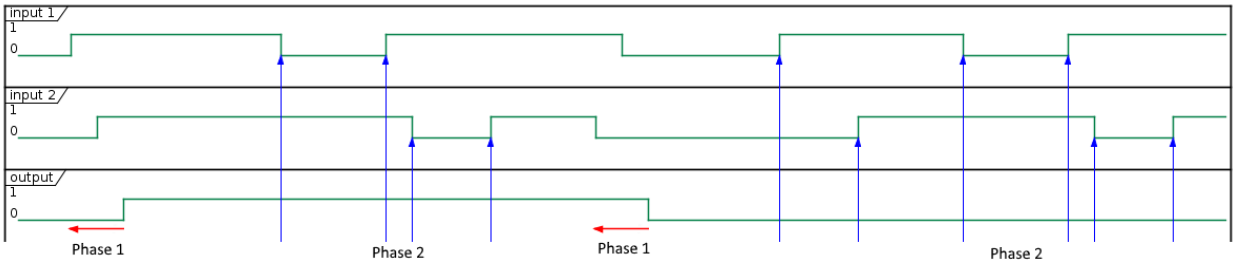

Figure 4.6: Conflicting Patterns

```
Phase 1:
input_1: 1 | input_1_t: 1 | input_2: 1 | input_2_t: 1 | light: 1
input_1: 0 | input_1_t: 1 | input_2: 0 | input_2_t: 1 | light: 0
Phase 2:
input_1: 0 | input_1_t: 1 | input_2: 1 | input_2_t: 0 | light: 1
input_1: 1 | input_1_t: 1 | input_2: 1 | input_2_t: 0 | light: 1
input_1: 1 | input_1_t: 0 | input_2: 0 | input_2_t: 1 | light: 1
input_1: 1 | input_1_t: 0 | input_2: 1 | input_2_t: 1 | light: 1
input_1: 1 | input_1_t: 1 | input_2: 0 | input_2_t: 0 | light: 0
input_1: 1 | input_1_t: 0 | input_2: 1 | input_2_t: 1 | light: 0
input_1: 0 | input_1_t: 1 | input_2: 1 | input_2_t: 0 | light: 0
input_1: 1 | input_1_t: 1 | input_2: 1 | input_2_t: 0 | light: 0
input_1: 1 | input_1_t: 0 | input_2: 0 | input_2_t: 1 | light: 0
input_1: 1 | input_1_t: 0 | input_2: 1 | input_2_t: 1 | light: 0
```

The first 4 lines of Phase 2 indicate that the output should be 1, but the last 4 indicate, for the same values, that the output should be 0, presenting a clear contradiction.

So far, the expected output values have been 0 and 1. These values will remain as the possible results for the data of phase 1, but in order to make the distinction between it and the data from phase 2, a new possible result in the form of -1 is introduced. This way it's possible to distinguish between when the output device is to change state and when it's not.

4.4.5 More Information - Continuous Values

As seen in section 4.4.3, sometimes more information is necessary in order to have more detailed behavior. A light sensor was introduce to the example of fig. 4.2. Now when the presence sensor gets triggered and the luminosity sensor indicates there is not enough light, the light is turned on.

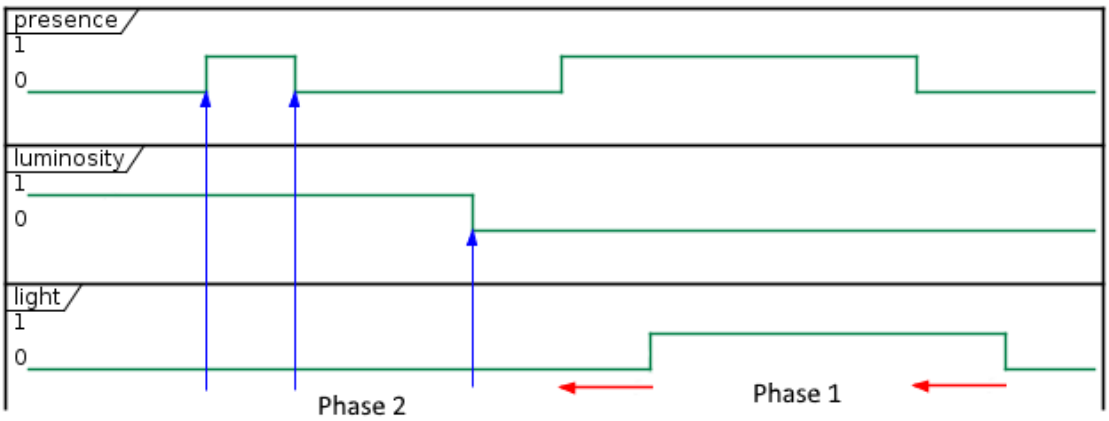

Figure 4.7: Continuous Values

Continuous values work differently from binary values. Contrary to binary values, it does not matter that a continuous measurement transitioned to a certain value, merely that it is currently at that value. As such, the transition feature does not apply, giving the data:

```
Phase 1
presence: 1 | presence_t:1 | luminosity: 0 | light: 1
presence: 0 | presence_t:1 | luminosity: 0 | light: 0
Phase 2
presence: 1 | presence_t:1 | luminosity: 1 | light: -1
presence: 0 | presence_t:1 | luminosity: 1 | light: -1
presence: 0 | presence_t:0 | luminosity: 0 | light: -1
```

Handling continuous and binary values differently implies the need to distinguish between them. Therefore, the system will need to provide the user the option to select whether the device's values are continuous or binary. If they are continuous, the user will be able to select the minimum and maximum expected values and discretization interval.

Due to the nature of continuous values such as temperature, luminosity and humidity, it is not expectable to have data for all possible values. As such, discretization of these continuous values is needed. The project will make use of the simplest discretization method, equal width interval binning [DKS95].

This method divides the values into k equally sized bins, where k is a parameter provided by the user. Each continuous feature is bounded by a max and min value, which allows the calculation of the size of the bin with:

$$\delta = \frac{max - min}{k} \tag{4.1}$$

The max and min values are also supplied by the user. As an example, a user wants to set up a luminosity sensor that gives values from 0 to 9, 0 being the darkest and 9 the brightest. From the knowledge he already has of the sensor, the user knows that he needs light whenever the sensor register a value of 4 or below, allowing the system to take advantage of further apriori knowledge, and giving it the necessary information to discretize all the possible values into just 0 or 1, by selecting a k value of 2.

4.4.6 Additional Considerations

In section 4.4.2 a transition feature for binary devices was introduced. However, its usefulness proved to be less than previously thought. Ultimately the transition feature allows the distinction, for example, of the moment a person enters a room and the person simply being in the room which is unnecessary. As such this feature was removed, which was the primary cause for the distinction between binary and continuous devices. Binary values can be seen as continuous values with a chosen k of 1, 0 as min and 1 as max. As such the distinction between these types of devices was also removed.

The causality window, introduced to allow for the transition feature, is still being used for reinforcement, as seen in section 4.7.2 and other cases, as seen in section 4.7.2.1.

In section 4.4.4, a new possible result for the data from Phase 2 introduced in the form of -1, indicating that the state of the output device should remain the same. However, the distinction can also be made by adding a feature with the most recent value of the output device, in relation to that entry. With this adjustment, the data from fig. 4.6 would be:

```
Phase 1:
input_1: 1 | input_2: 1 | light_p: 0 | light: 1
input_1: 0 | input_2: 0 | light_p: 1 | light: 0
Phase 2:
input_1: 0 | input_2: 1 | light_p: 1 | light: 1
input_1: 1 | input_2: 1 | light_p: 1 | light: 1
input_1: 1 | input_2: 0 | light_p: 1 | light: 1
input_1: 1 | input_2: 1 | light_p: 1 | light: 1
input_1: 1 | input_2: 0 | light_p: 0 | light: 0
input_1: 1 | input_2: 1 | light_p: 0 | light: 0
input_1: 0 | input_2: 1 | light_p: 0 | light: 0
input_1: 1 | input_2: 1 | light_p: 0 | light: 0
input_1: 1 | input_2: 0 | light_p: 0 | light: 0
input_1: 1 | input_2: 1 | light_p: 0 | light: 0
```

With the introduction of this feature, the number of overall features is reduced alongside the possible results. There are no conflicts in the data, which is also more readable.

4.5 Prediction

One of the main objectives of this project is the ability to use the patterns learned in order to control devices. The messages that arrive to the system are according to the MQTT protocol described in section 5.1.2, and are composed of a topic and a payload. The payload contains the value of the input, and the name is extracted from the topic.

Prediction can be broken down into the following steps:

1. When the user interacts with an input device new data reaches the system;
2. Database information is updated;
3. Most recent values of the devices are retrieved from the database;
4. Prediction data is created and given to the selected algorithm for a prediction:
   (a) If in Automatic Mode, the prediction is sent to the device;
   (b) If User Authorization is required (section 4.6), the prediction is sent to the user, as described in section 4.7.1.

Data creation for training and prediction is very similar. Where for training all data from a given time interval is necessary, prediction only needs the most recent values of each input device. When information reaches the system, it's first inserted into the database in order to keep it updated and then the most recent values of each device are retrieved and formatted, with values being discretized:

```
light_1_p:0 | lux_1:0 | presence_1:1
```

Due to the use of the previous value of the output device, that data needs to be saved as well. This creates a feedback loop as detailed in section 4.7.3.

4.6 User Authorization

Depending on the type of output device being used, it might be relevant to provide an option for when the user wants to be notified and allow or refuse the decision of the system. This feature can also used to gather user feedback on system decisions, in order to allow for reinforcement.

4.7 Reinforcement

Another important aspect of the project is to provide the user with a way to aid the system make decisions, via reinforcement.

Reinforcement is done in two ways, by making use of the multi armed bandit approach described in section 2.3.4.2 using the upper confidence bound formula and by analyzing user behavior during training data creation. However, due to the need for the system to save its own decisions, it also ends up reinforcing its own behavior.

4.7.1 Multi Armed Bandit with Upper Confidence Bound

The prediction process, alongside the reinforcement process using MAB, can be divided into 4 steps:

1. New data from an input device reaches the system, is formatted into prediction data, and a prediction is obtained, as described in section 4.5.
2. The decision made by the system is rewarded with a value of 1 point.
3. Using the UCB formula, a value is selected to be presented to the user.
4. The user can accept or reject the decision of the system.
   (a) Accepted decisions receive a reward of 3 points, and are sent to the output.
   (b) Rejected decisions receive a reward of 0 points, and are not sent to the output.

These decisions are then saved or updated in order to be used again in the future. Since no assumptions can be made regarding all the possible results of a combination of features, the training data is iterated over in order to get all the possible results, which allows the decisions to be initialized.

4.7.2 Via User Action

Reinforcement can also be made when the user or the system made the wrong decision, and then the user decided to correct it, introducing another a more natural or intuitive reinforcement, where the training data is corrected during its creation.

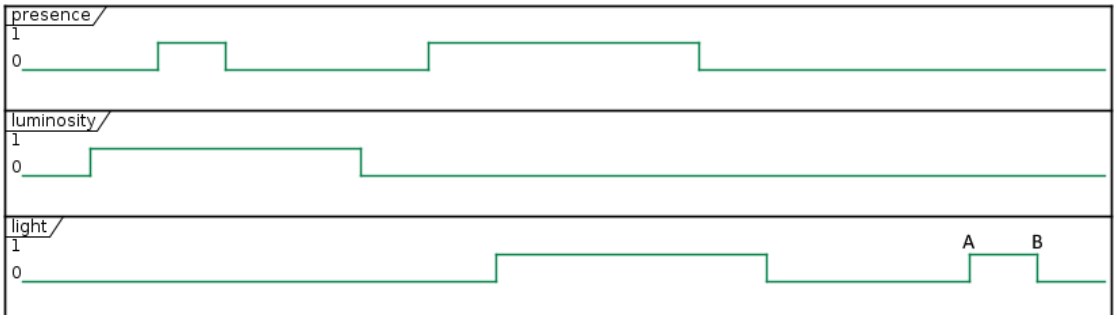

Figure 4.8: Reinforcement Via User Action Example

In the case of fig. 4.8, the system or user turned on the light and then the user proceeded to turn it off for whatever reason.

In order for this process to be applied, there is need for more information to be processed in Phase 1. Besides getting the most recent values of other devices, all other entries of the output device itself inside the causality window will also be processed. The window will search not only backwards, but also forwards.

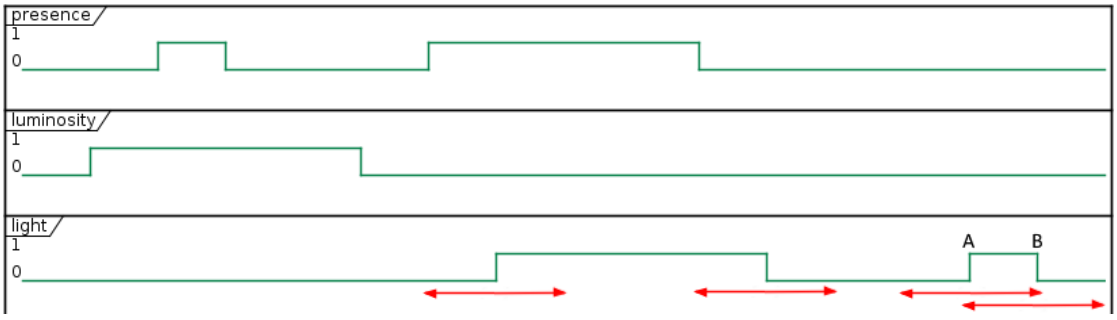

Figure 4.9: Reinforcement Via User Action Example w/Windows

In this case event A, which may exist as an error because of the user or the system, is being corrected by the user. Instead of being ignored as an error, when analyzing the data an opportunity arises to correct the behavior of the system.

Output entry A's causality window will detect entry B. Considering entry B has a different value, the training data entry that would have originated entry A is corrected:

```
Before:
presence: 0 | light_p:0 | luminosity: 0 | light: 1
After:
presence: 0 | light_p:0 | luminosity: 0 | light: 0
```

Then, when it's time to process entry B, the training data will be the same as for entry A. This way, not only is A corrected by B, but the training data ends up with two instances of when to not turn on the light instead of one, lending more weight to the data.

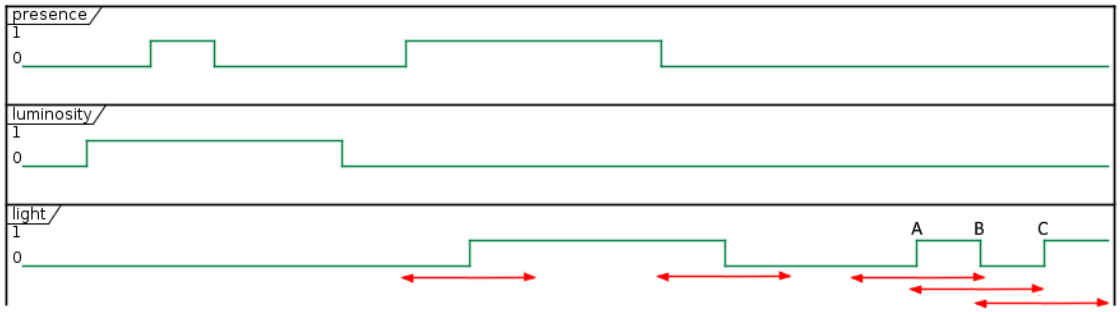

Figure 4.10: Reinforcement Via User Action Example 2

In the situation of Figure 4.10, A is being corrected by B, but then B is being corrected by C. This time A will result in incorrect data. This may be problematic if it occurs with enough frequency for the learning algorithms to set it as a rule. However, in this case, the data still ends up with two correct situations versus one incorrect one.

With the introduction of this reinforcement methods, the following situations involving the entries from the output device are taken into account:

1. There are no other entries in the causality window, there being no reinforcement from the user;

2. There is one entry before the current one - Current entry is taken as reinforcement by the user, current entry is unaffected;

3. There is one entry after the current one - the current entry is being corrected;

4. There is one entry before and one after - it proceeds as situation 3.

In any other combination, the system assumes there are too many entries for the causality window defined by the user and the entry ignored as a possible error.

4.7.2.1 Effect on the Training Data - Effect and Cause

With the causality window being extended not only backward but also forward, it resolves an issue that hasn't been dealt with so far, that the system has advanced with the assumption that devices are activated in consequence of actions that happened before. However, it is also necessary to assume that events might happens the other way around.

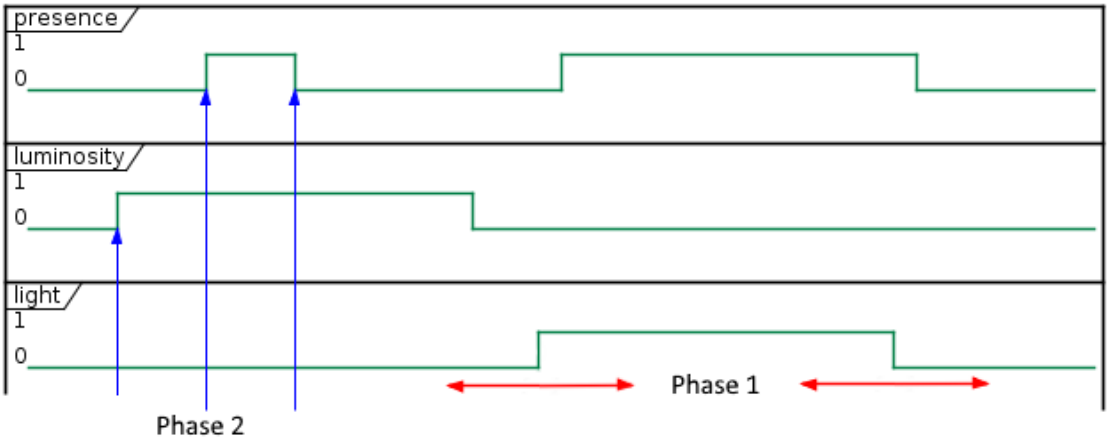

Figure 4.11: Effect and Cause

In the example of fig. 4.11, the light is turned on before the user sits down, which is a case not accounted so far. While before only the closest event in time that happened before the output device event, With the information gathered from the causality window, the system is now taking into account the closest events in time, whether they happen before or after.

4.7.3 Feedback Loop

Having information updated is important since the most recent values of the devices are necessary to create the prediction data, and will most likely result in error if the information is incomplete.

As such, the system needs to save the output information, which reaches the system when the user interacts with a input device, but also when the system executes predictions. This means that the system ends up reinforcing its own behavior because it saves its own decisions. Depending on the accuracy of the system, this might be undesirable, since it might end up creating data where the value of the output device is wrong.

Since the user can give feedback at a later time, the system must save the decision with a different timestamp. In order to preserve causality this data is being saved half a time window after the prediction is made.

4.8 Higher Level Information

So far, the design process has been dealing with numerical values, but these are not the only kind of values that hold significance. Using numerical values can hold back the predictive potential of the system by making situations harder to distinguish. For example, a presence sensor that only detects whether or not something is present will always indicate that the light is to be turned on, even if the user doesn't need it.

What if there is a sensor that, using computer vision techniques, can tell if it's a pet instead of a human? What if there is a complex sensor, or array of sensors, that can tell if the user is watching television or reading, and the light is not to be turned on if he is simply watching television?

With higher level information at the system's disposal, it would be easier to make predictions in more complex scenarios without being forced to think only in terms of zeros and ones.

If a non numeric value is detected, it will be used as is, since the system does not know anything about the context of its use and no assumptions can be made regarding this kind of information.

4.9 Learning Algorithms

For training and prediction, the user will be able to select from 3 algorithms, and have the option to use them as an ensemble. The various parameters for the algorithms will also be adjustable.

4.9.1 Random Forest

The first considered algorithm, due to it's simplicity was Random Forest. This established the default format of the training and prediction data. The project will be making use of the *random-forest-classifier* [rf-] Nodejs module. This library takes objects where the properties are the features and trains the algorithm using one of the features as target. The final prediction value returned does not need further processing.

4.9.2 Feed Forwards Neural Network

The second considered algorithm was a feed forward neural network, making use of the *limdu* [lim] Nodejs module. This library accepts pairs of inputs and output, and the features, as well as the output are values between 0 and 1. The appropriate formula needs to be used in order to convert from one min-max set to another:

$$\text{Result} = \frac{\text{Input} - \text{InputLow}}{\text{InputHigh} - \text{InputLow}}(\text{OutputHigh} - \text{OutputLow}) + \text{OutputLow} \tag{4.2}$$

The resulting prediction will also be a number between 0 and 1, which is not directly usable, and two thresholds to be defined by the user are introduced, an upper and a lower one. If the prediction is between thresholds, the algorithm is not sure of the result and it returns the previous value of the target device, indicating its state should not be altered. Otherwise it returns a 0 if below the lower threshold or a 1 if above the upper threshold.

4.9.3 Association Rules

The last technique used was the creation of association rules. Usually, machine learning algorithms are fed information in order to make prediction on new, never seen before data. However, within the context of this project, new data is not expected, but that the user keeps up with his usual patterns.

With that in mind, association rules stand out as a useful data mining technique, bringing multiple advantages: it's simple to use, the created rules are readable to humans and allow the user to understand why the system is making the decisions and the rules can be saved and loaded, preventing re-training when the system is shut down or restarted.

The use of this technique will create a set of rules, with a antecedent and consequent, where the antecedent is the set of features and the consequent the expected value for the output device, such as:

```
[presence_1:1, light_1_p:0] => light_1: 1
```

For prediction, data will be formated into a antecedent, and matched to the set of rules. In case this antecedent finds no matches, it will return the previous value of the output device, effectively maintaining its state, or doing nothing. For this reason, the training data only needs to go through the first phase of analysis described in section 4.4.3.

At first, the system made use of the Apriori algorithm via the *apriori* [apr] Nodejs module. However, this algorithm presents a disadvantage taking into account the context of the problem. It calculates the support for every combination of features, while the system is only interested in a specific kind of antecedent, one that contains all features from the input devices, and one kind of consequent, one that contains the expected value for the output device.

Considering that the training data is already organized in a rule like format, the application of the Apriori algorithm would effectively result in culling noise, data that is not frequent enough to be considered a pattern. This would start taking too much time when the number of input devices increases because the number of features would also increase.

As such, in order to create the association rules, the system checks the support for each entry of the training data, calculating the percentage of times that entry appears in the total amount, and culls entries below a certain user-defined threshold, turning the rest into rules.

4.9.4 Ensemble

Ensemble can sometimes improve the predictive power by using the predictions of multiple models and it's easy to implement. The ensemble method will be making use of the three algorithms presented thus far and decide the final result via majority vote.

4.10 Testing

In order to validate the system, testing capabilities will be required. Considering the aspects described so far, testing will encompass running tests for the designed scenarios and testing the reinforcement aspects of the system. For the scenarios, synthetic data will be created using a script and imported into the database.

In order to see the effect of the introduction of the data of phase 2 and data repetition, it will also be possible to chose whether or not to just use phase 1 during training data creation and to remove duplicate entries from the training data.

4.11 Conclusions

This chapter presented the design decisions involved in the IM2HoT project. In addition to the user options listed in section 4.2, after the decomposition of the system and analysis of the necessary steps, the following options need to be added:

- Select the algorithm to be used, as well as adjust algorithm specific parameters
- Enable ensemble decision mode
- Enable use of Phase 2 and removal of duplicate entries in the training data
- Enable testing mode and selection of the test file
- Enable user authorization mode

The system will be making use of Random Forest, Feedforward Neural Network and Association Rules to predict the behavior of the output device, as well as an ensemble mode using majority vote. Even though the neural network cannot deal with higher level information as described in section 4.8, random forest and in particular association rules should be able to.

The creation of the training data will involve two phases. Phase 1 will analyze the data using the database entries of the output device as anchor and analyze the situation around them, and phase 2 will analyze the remaining entries in order to gather as much useful data as possible.

The presented decisions cover all of the expected contributions delineated in section 3.1. Details regarding their implementation will be addressed in chapter 5

Chapter 5

Implementation

This chapter will cover relevant implementation details of the features described in chapter 4, starting with the technological choices for the project. It will then present the main implementation details of the multiple features explored in chapter 4.

5.1 Technological Choices

From the objectives detailed in section 3.1, the following features are required for the IM2HoT project:

- User Interface
- Communication with devices
- Data storage

This section details the technologies chosen for the development of the project: Node-Red for user interface, MQTT for communication with devices and InfluxDB as database.

5.1.1 Node-Red

Due to the nature of the proposed solution as an IoT system with a reinforcement mechanic and several user options, user interaction is an important aspect of the project.

Users will need to define what the input and output devices are, multiple options regarding the training of the models, such as what devices are to be controlled, the time interval when it comes to the training data, when the system is in control of the devices and to allow reinforcement learning.

Node-Red [NR17] is a visual programming tool that uses flow-based model to build IoT applications. In this programming model, the logic of the application is expressed as a directed graph, called flow, where each node can have inputs and outputs. The nodes which only have outputs or inputs represent the start and end of the flow. Each node processes the input information and

outputs it for the downstream nodes. The nodes are independent, their execution does not affect the behavior of other nodes, making them reusable and portable [Gia15].

Node-Red provides a graphical interface that allows the user to drag and drop nodes into a canvas. This kind of logic can be more intuitive for non-programmers to understand, and allows users to quickly move between design and implementation, reducing development time [BL14].

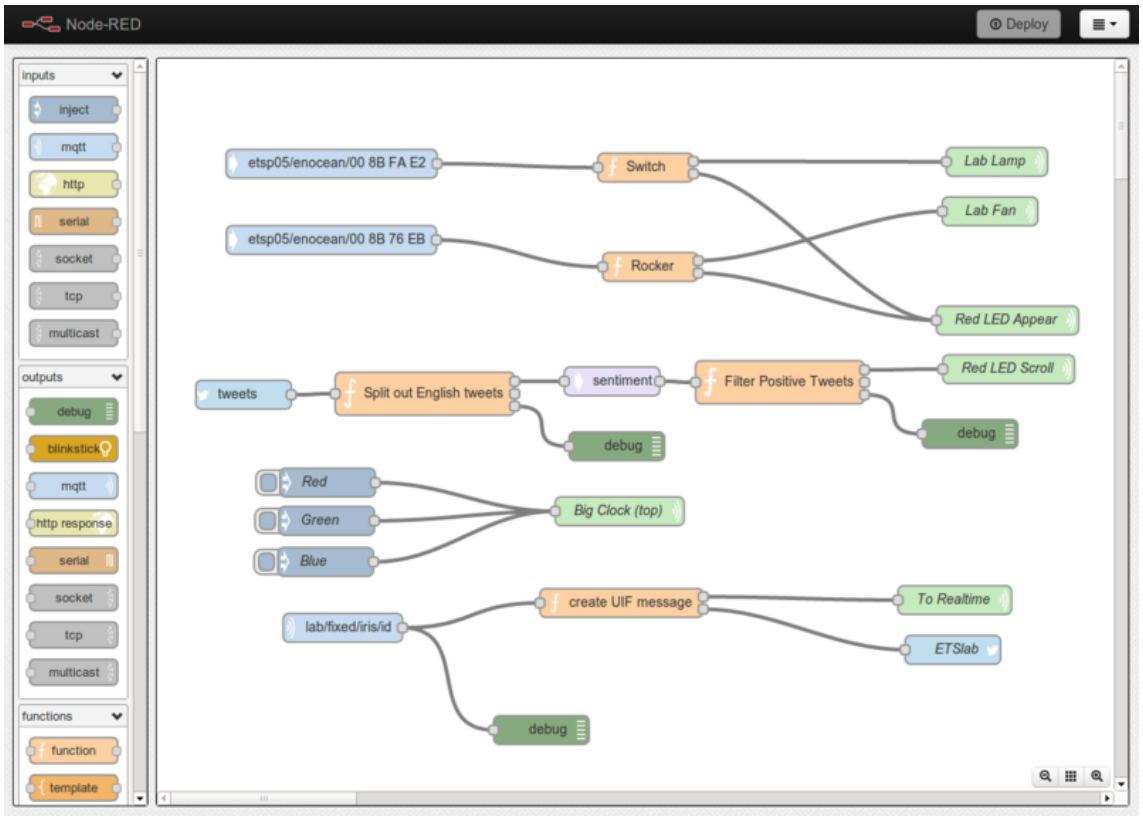

Figure 5.1: Node-Red Flow Example

The use of visual data flow programming languages is also present in other domains such as high performance parallel computing, music, toys and industrial applications [BL14].

Beyond user experience, another advantage Node-Red possesses is a large community of users contributing new nodes and flows. Nodes can be implemented by an HTML file, which implements the aspect of the node in the canvas and a JavaScript file, which implements the logic. The flows themselves can be represented textually, allowing them to be easily exported and imported. Even if a node used in one instance is not available in another, a placeholder node is shown, indicating to the user where the error lies, making debugging easier.

The expected final result of the IM2HoT project is then a Node-Red node, where the user will be able to connect the input and output devices and set the necessary options.

5.1.2 MQTT

This project also requires a mean of communicating with devices, in order to control them and gather information about their use.

MQTT is a simple and lightweight messaging protocol, with a publish/subscribe architecture designed to be open and easy to implement, being capable of handling thousands of remote clients with a single server [Kee12].

The publish/subscribe architecture is well suited for the distributed nature of IoT applications, which demand more flexible communication models. In this model there are two sides, the publishers and the subscribers. Subscribers can express their interest in an event, and are notified when a publisher generates the same kind of event.

The strength of this model lies in the decoupling of publishers and subscribers. which removes explicit dependencies, increasing scalability [EFGK03].

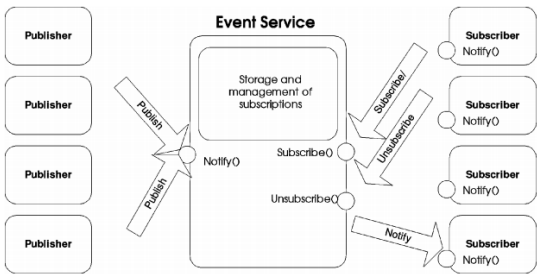

Figure 5.2: Publish/Subscribe Architecture example

In MQTT, the publisher has the responsibility of classifying their messages by using topics. Topics define under which category may the content of the message be categorized. These topics are organized hierarchically into trees, where the '/' character is used to create subtopics, for example *home/livingroom/sensor1* and *home/bedroom/ceilinglight*.

To ensure message delivery MQTT defines three quality of service (QoS) levels. An higher QoS level ensures more reliable message delivery, but might consume more bandwidth or subject the message to delays. MQTT also has the ability to retain messages. The server keeps the messages even after sending it, so that any new subscribers can receive these retained messages [Kee12].

MQTT, when compared to other messaging protocols, offers various benefits [Kee12]:

- Simplicity - MQTT was made open so that it can easily interact with other applications;
- Publish/Subscribe Model - These models allows decoupling of sender and receiver. As such, publishers do not need to know who or even if is receiving their messages, and vice-versa;
- Minimal Maintenance - Features like automated message storage and retransmission minimize the need for on-the-fly maintenance;

- Limited on-the-wire footprint - MQTT keeps data overhead to a minimum on every message;
- Continuous session awareness - By being aware of when session terminate, appropriate action can be taken;
- Local message processing - MQTT assumes that remote devices have limited processing capabilities;
- Message persistence - With QoS, the publisher can ensure delivery of important messages;
- Agnostic regarding data types - MQTT does not require any particular format for the content of the messages.

Node-Red also provides native MQTT capabilities in the form of input and output nodes. The input nodes connects to a broker and subscribed to a specified topic. The output node publishes to either a specified topic in the node or the topic of the incoming message.

The relevant aspects of the MQTT protocol for the project is the format of its messages. From the messages that reach the node two pieces of information need to be extracted: the value of the input, which is in the payload field, and the name of the device, which is considered to be the last subtopic in the topic of the message. This subtopic must match the *device_id* mentioned in section 4.3.

5.1.3 InfluxDB

The collection of multiple kinds of data, such as server metrics, application monitoring, network load and sensor data over time is done with time series.

A time series is a sequence of values, each with a timestamp indicating when the value was recorded. A time series database is a database that provides a way to store time series data and then retrieve it for a selected time range with particular efficiency. Time series data can also be useful in the detection of patterns [DF14].

InfluxDB is an open-source time series database, optimized for time series data, with a format as exemplified below:

```
<measurement name>,<tag set> <field set> <timestamp>
cpu,host=serverA,region=uswest idle=23,user=42,system=12 1464623548s
```

When compared with OpenTSDB [Ope], a popular time series database and alternative choice, InfluxDB outperformed OpenTSDB in data ingest performance (5x better), on-disk storage requirements (16.5x better) and mean query response time (4x faster) [Inf16b].

When compared to MongoDB [Mona], a popular NoSQL database natively supported by Node-Red and possible alternative to handle time series data, InfluxDB outperformed MongoDB in data ingest performance (27x better), on-disk storage requirements (84x better) and was comparable in query performance, being better in low concurrency [Inf16a]. It is worth nothing that

while MongoDB is not a time series database, it is promoted for its use for time series workloads [Monb]. Additionally, InfluxDB has an easy to use SQL-like language, providing a familiar approach to queries, as exemplified:

```
SELECT value FROM series WHERE time > '2015-04-15'
SELECT value FROM series WHERE time > now()-1h
SELECT value FROM series WHERE time > now()-1d LIMIT 100
```

5.2 Implementation Details

This section details the most relevant implementation details of the project.

5.2.1 Database Access and Information Retrieval

The IM2HoT node retrieves information from the database using a query language similar to SQL. InfluxDB does not support the notions of joins, so accessing each series requires its own query. The example below shows the necessary queries to access the information pertaining devices *presence_1*, *presence_2*, *light_1* and *lux_1*. The results are ordered by time and grouped by device_id so that they are easier to search.

```
SELECT * FROM event WHERE device_id='presence_1'
OR device_id='presence_2' OR device_id='light_1'
AND time >= '2017-08-15T00:00:00Z'
AND time <= '2017-10-15T00:00:00Z'
GROUP BY device_id ORDER BY time DESC

SELECT * FROM luminosity WHERE device_id='lux_1'
AND time >= '2017-08-15T00:00:00Z'
AND time <= '2017-10-15T00:00:00Z'
GROUP BY device_id ORDER BY time DESC
```

Each call to the database is asynchronous, which means that calls would need to be nested within each other, which is an impossibility since number of series used per node is not known beforehand.

This issue was solved with the use of Node's supported async/await capabilities, which allows asynchronous calls to be made in order, as if they were synchronous. Each query also requests the data in a descending order by time, which facilitates getting the most recent values of each device.

An advantage Node-Red possesses is the large number of available nodes, easily downloaded and installed. However, this would create an outside dependence on an extra node, which is undesirable. As such, the necessary code from the *node-red-contrib-influxdb* [nod] contribution, which makes available extra nodes to retrieve and save information in an InfluxDB database, integrated into a configuration node internal to the IM2HoT system.

Configuration nodes are internal to the system and are create when an IM2HoT node is placed in the canvas. These nodes provide configuration options that are transversal to all IM2HoT nodes. Multiple configurations can be saved, which allows each node to access a different InfluxDB database, if the user so wishes. The configuration options related to the database can be seen in fig. 5.3. Fig. 5.4 shows the name and configuration node settings of the IM2HoT node. The user can create a new configuration node with new settings or edit existing ones. Fig. 5.5 shows the date and time options, where the user defines the time window for search and the duration of the causality window.

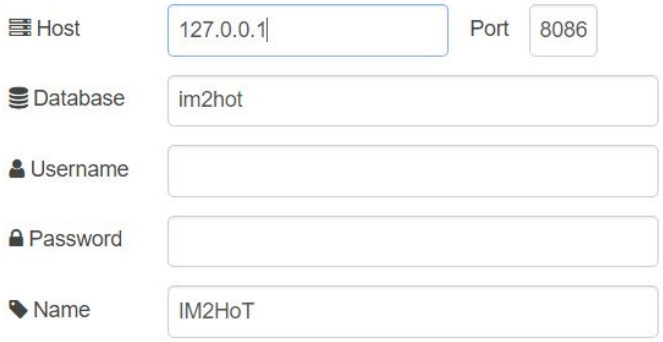

Figure 5.3: InfluxDB Configuration Node Settings

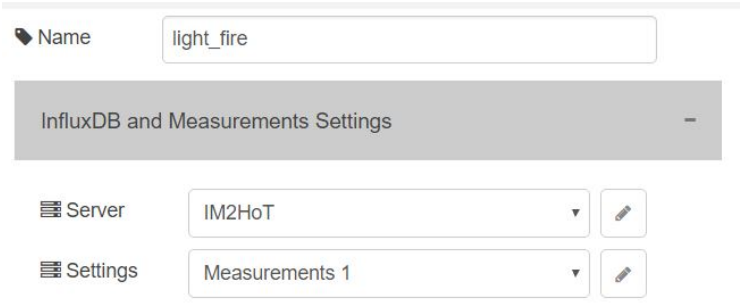

Figure 5.4: IM2HoT Configuration Node Settings

5.2.2 Input Devices

The behavior of the output device will be decided by a set of input devices, which are to be selected by the user. Ideally, the user would only need to connect the devices to the node in the Node-Red canvas for the node to automatically create a list.

However, in Node-Red, nodes do not possess the capacity to know what other nodes are connected to it, as this would violate the principle that each one does not need to concern itself with those that surround it, but only with the information it receives.

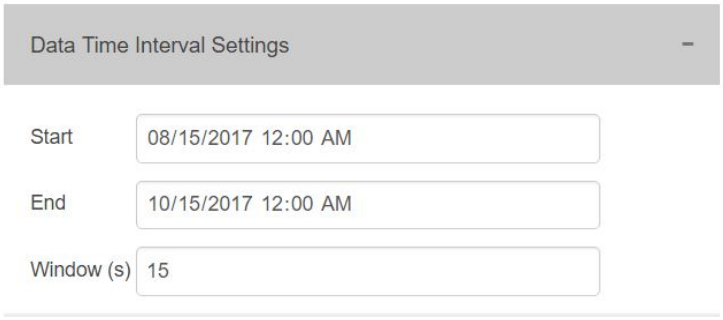

Figure 5.5: IM2HoT Date and Time Settings

To this end, the node makes use of the container in fig. 5.6 and the ability Node-Red has to create addable objects. These objects are defined in the html file of the node.

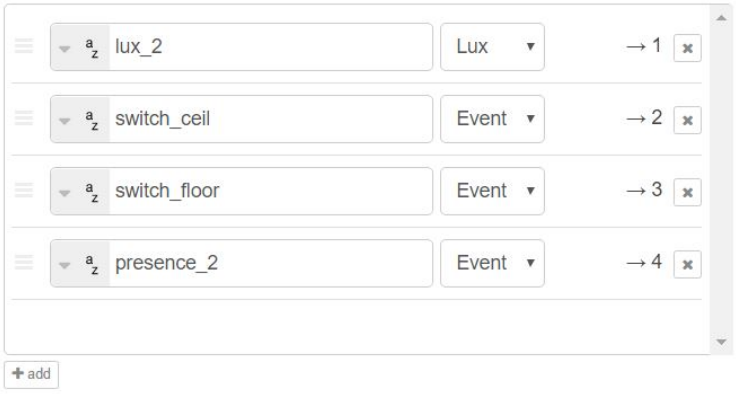

Figure 5.6: Output Device Container

Output devices are defined by a name, which must match the database's *device_id* field, and a user defined type. The creation of these custom types can be found in section 5.2.2.1. The order that the input devices appear in on the list has no bearing on the system.

5.2.2.1 Custom Device Types

Information in InfluxDB is organized into series. Instead of forcing the user to setup a specific number of series with specific names for types of variables, it's better to allow him to define which series and variables to use. To this end, the system provides the capacity to add new measurements in a configuration node, whose settings can be accessed from any IM2HoT node.

These measurements are defined by a name, series, type and number of intervals, min an max values for discretization. After creation the name will show up in the dropdown menu of the input device list of the node, as shown in Fig. 5.6. The series will be used in query building to access InfluxDB, and the interval and min and max values for training data creation, as described in section 4.4.5.

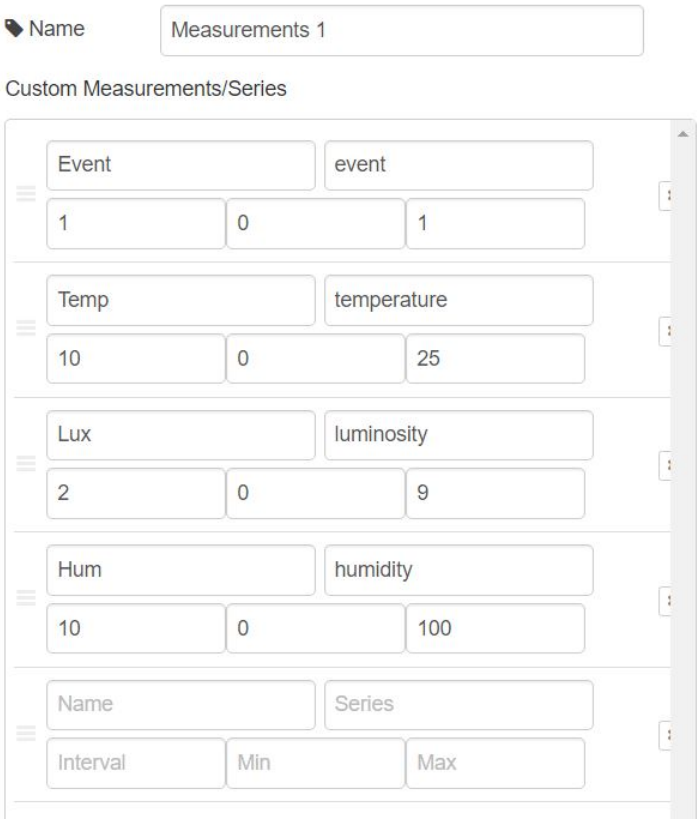

Figure 5.7: Custom Measurements/Series

5.2.3 Training

As mentioned in section 4.9, the user will be able to select the learning algorithm, as well as algorithm parameters. In order to see the effect that the training data creation phases have and the repetition of data, extra options to select which phases to use and whether or not to remove repeated entries from the data were added. These options can be seen in Figure 5.8.

The system trains the selected algorithm, or all of them in case of Ensemble, when a message arrives with the payload *training*. This message can come from an MQTT node set up by the user, but Node-Red also provides an injection node that allows users to manually or automatically trigger messages with the same format of topic and payload.

5.2.4 Prediction

Predictions are executed when a new message from an input device reaches the IM2HoT node and training isn't started.

This message contains a payload from which the name of the device is extracted alongside the value, which is then inserted into InfluxDB. After that the most recent values of the input devices are queried for, the prediction data is created and given to the selected algorithm.

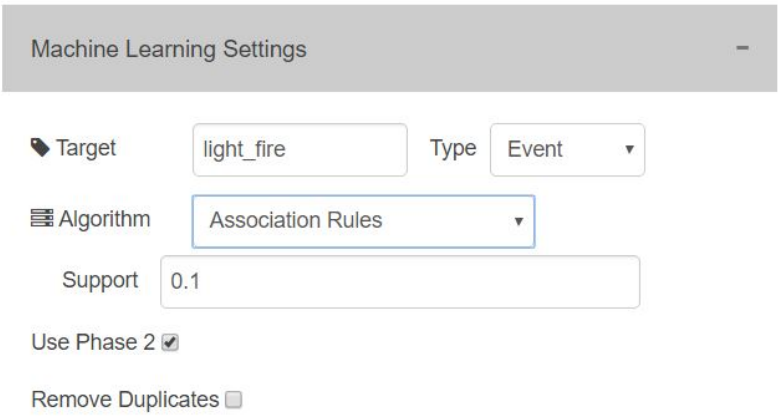

Figure 5.8: Machine Learning Settings

In order to have the database updated, the output device needs to be connected to the node alongside the input devices for its data to be recorded when the user interacts with it. When data arrives, if it belongs to the output device, it is saved to the database, with nothing else being done after.

5.2.5 Reinforcement

The involvement of the user in the learning process of the project comes in the form of reinforcement, where the user is notified of the decision of the system, and can accept or reject it.

5.2.5.1 Slack

Due to ease of integration and use, Slack was the platform of choice to implement a simple version of the reinforcement mechanic. In its channels, Slack allows the integration of apps designed to help users with their work.

With the use of interactive components, users can receive messages with which they can interact, which is used to present the decisions of the system to the user. Messages are sent to Slack via a POST request to a webhook URL defined by the app, and result in a message with interactive components, as seen in Figure 5.9.

Upon user interaction, the app sends a POST request to an URL that needs to be defined in Slack's API platform. Fortunately, Node-Red allows nodes to define their own REST endpoints. Since all interactive components of the same app send POST requests to the same URL, all IM2HoT nodes have to share the same endpoint URL, and make the distinction to which node the response is meant to internally, something the system is not currently doing.

5.2.5.2 Saving User and System Rewards

All of the decisions and rewards described in section 4.7.1 need to be saved in order to be used in future iterations. Unfortunately, InfluxDB does not support the notion of update, so the decisions

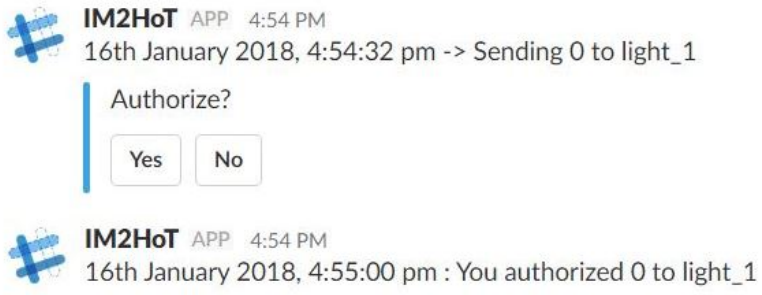

Figure 5.9: Machine Learning Settings

are instead saved in a JSON file. Each IM2HoT node has its own file. Nodes in Node-Red have their own unique identifier, but if these were used to identify the files, they would be inaccessible once the node was deleted, even if the replacement node used the same input and output devices. As such, each file is identified by a concatenation of the input devices organized alphabetically plus the output device, which uniquely identifies the node, such as:

```
lux_1presence_1light_1.json
```

To be uniquely identifiable, each decision is defined by the combination of the output device, the features and their values. Since there is no apriori knowledge of the number and format of possible decisions (e.g: numerical or string), the system first iterates over the training data to collect all possible results and initialize the decisions. As with the rules created by the Association Rules method, this JSON file can also be loaded every time Node-Red restarts.

```
[{"light_1light_1_p.0lux_1.0presence_1.1":
    {"0":{"count":1,"weight":0,"value":0},
     "1":{"count":3,"weight":1.66665,"value":1}}},
 {"light_1light_1_p.1lux_1.0presence_1.0":
    {"0":{"count":3,"weight":1.66665,"value":0},
     "1":{"count":1,"weight":0,"value":1}}},
 {"light_1light_1_p.0lux_1.1presence_1.1":
    {"0":{"count":3,"weight":1.66665,"value":0},
     "1":{"count":1,"weight":0,"value":1}}},
 {"light_1light_1_p.0lux_1.0presence_1.0":
    {"0":{"count":3,"weight":1.66665,"value":0},
     "1":{"count":1,"weight":0,"value":1}}},
 {"light_1light_1_p.1lux_1.0presence_1.1":
    {"0":{"count":1,"weight":0,"value":0},
     "1":{"count":3,"weight":1.66665,"value":1}}}]
```

Figure 5.10: Decision JSON File Example

The id of each decision is a concatenation of the features and their values, ordered alphabetically. This allows the id to be easily created and compared. The possible results are keys to their

own information, which composed of *count*, the number of times they were chosen; *weight*, the average of the rewards by system and user; and *value*, which is the value of the result, to be easily accessible without need to iterate over the keys.

In these decision the feature values are already discretized. That means that if the discretization interval changes there might be more or less decisions to take into account, and as such the information is no longer valid, and must be reset.

5.2.6 Testing and Other Options

Testing options and capabilities are an invaluable part of the project, since they allow for experiments and validation of the system. Testing options, alongside others deemed relevant, were placed under *Mode Settings*, seen in fig. 5.11.

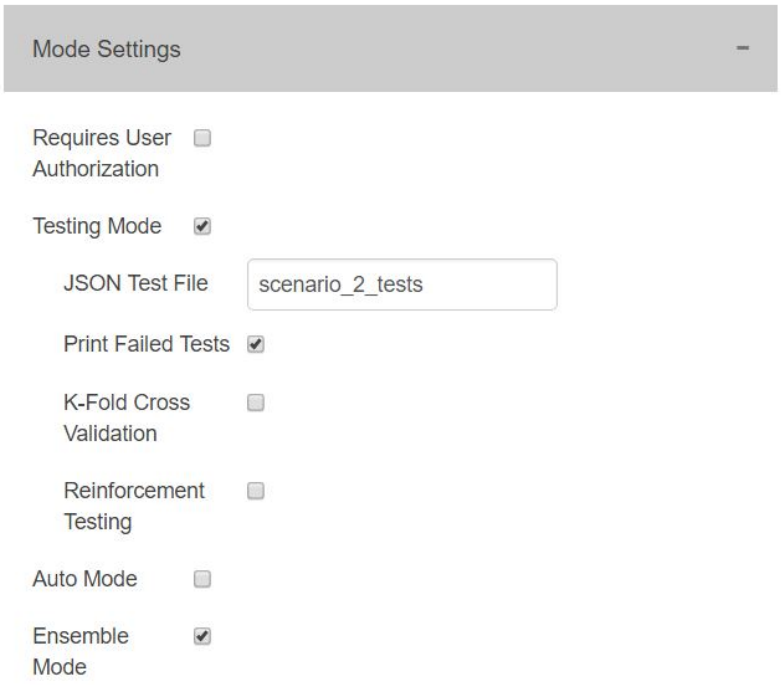

Figure 5.11: Testing and Other Options

Requires User Authorization enables reinforcement by the user by sending the decision to Slack, *Auto Mode* enables prediction once new data arrives to the node and in *Ensemble Mode* all algorithms are trained and prediction is made via majority vote, while *Testing Mode* opens up multiple other options. Enabling *Testing Mode* and *Auto Mode* are not compatible with each other and cannot run at the same time, as well as *K-Fold Cross-Validation* and *Reinforcement Testing*.

5.3 Final Result - The IM2HoT Node

Taking into account the use of Node-Red, the final result of the IM2HoT project is a single node, where all the devices will be connected as can be seen in fig. 5.12.

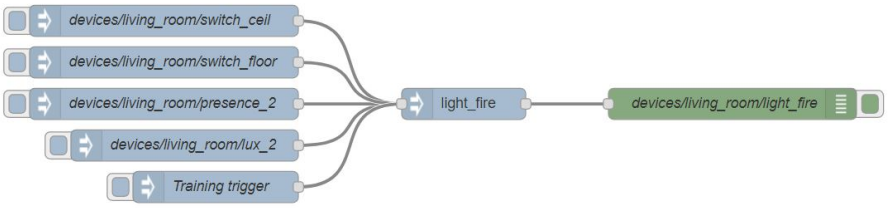

Figure 5.12: The IM2HoT Node

Fig. 5.13 shows multiple input devices, connected to three IM2HoT nodes, each controlling one output device. On the right are presented the properties of a node, and on the left the IM2HoT node can be seen inside its own category.

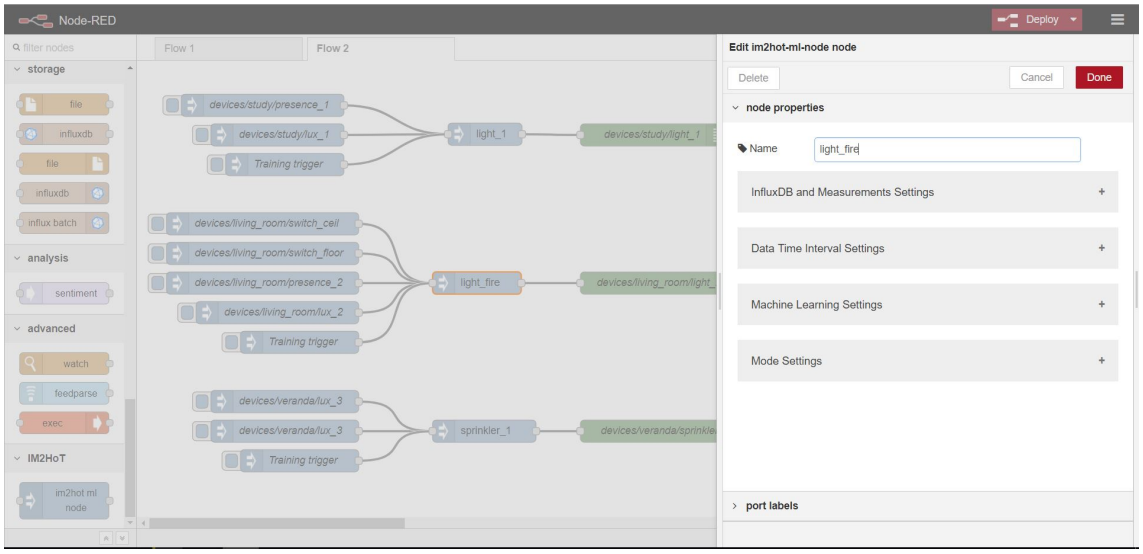

Figure 5.13: IM2HoT Flow

5.4 Conclusions

This chapter presented the most relevant implementation details regarding the design decisions presented in chapter 4, alongside the technological choices for the project.

All features presented in chapter 4 regarding user options, database access and information retrieval, training data creation, prediction and reinforcement were successfully implemented, and will be tested in chapter 6.

Chapter 6

Validation

This chapter will present the testing method and scenarios used to validate the system.

Each algorithm will be tested in 2 scenarios, described below, using the k-fold cross-validation method as well as tests created for each scenario containing feature combinations and the expected results. These tests can be found in annex A.

Considering the multiple phases and options described in chapter 4, each scenario will be tested using either phase 1 of the creation of the training data or both phases described in section 4.4, as well as with and without repeated entries in the training data.

K-fold cross-validation, running a test batch and reinforcement testing were implemented for all algorithms.

6.1 K-Fold Cross-Validation

The estimation of the accuracy of a classifier gives its future prediction accuracy and allows the choice of a classifier, or combination of classifiers, that better perform given a set of data.

In k-fold cross-validation, the dataset is split into k mutually exclusive subsets, or folds, of approximately equal size. The classifier is then trained and tested k-times, each time using one subset for validation, and the remaining as training data [Koh95]. The final estimate of accuracy is the average of the results of each iteration.

The k-fold results presented below are the average of 10 runs using 10-fold validation.

6.2 Data Generation

Data was generated with a custom script and imported to InfluxDB. The following test scenarios will make use of light and presence sensors, as well as switches for input devices. Even though each scenario is different, data for these sensors is created in the same way for each one:

- Luminosity is updated once an hour, and returns a value between 0 (darkest) and 9 (brightest).

- The presence sensor is updated when a presence is detected and returns a 1 when it detects a presence and 0 when no longer detects anything.

One week's worth of data was generated for each scenario.

6.3 Algorithm Parameters

The parameters for each algorithm are the same throughout the multiple tests, which are presented in table 6.1.

Table 6.1: Algorithm Parameters

Random Forest	
Trees	10
Feed Forward Neural Network	
Error Threshold	0.005
Max Train Iterations	20000
Learning Rate	0.3
Lower Threshold	0.2
Lower Threshold	0.8
Association Rules	
Support	0.1

6.4 Scenario 1

In this scenario, there is a lamp in an office, a presence sensor in a chair and a luminosity sensor that reads the exterior luminosity. The objective is to control the light using the two sensors.

The target behavior is as follows:

- If there isn't enough light and a person is in the chair, the light is turned on.
- If the light is on and there is no presence in the chair when it's dark, the light is turned off.
- In the remaining cases, the state of the light remains the same.

Daily data creation rules for this scenario are as follows:

- At 18h00, a presence is detected and the light is turned on, for a random duration between 15 and 45 minutes.
- There is a 50% change a person gets up during this interval, for a random duration between 1 and 5 minutes, and then returns.
- This routine happens again at 21h00.

- A person sitting down can happen before or after turning on the light.
- The interval between the first and second event is of random duration between 5 and 15 seconds.
- This routine happens once more at 17h00, during the weekends.

For this test, the discretization interval of the luminosity measure is set to 5, effectively dividing the range of values into bright (5 to 9) and dark (0 to 4), and with a time window of 15 seconds. In this scenario the user is never present in the chair with a luminosity of 5 and above, creating a gap in the information.

If a person gets up and the light remains on, the resulting data indicates that the light should remain on when it's dark and there is no presence, which creates a conflict with the situation when the same conditions indicate the light should be turned off. This conflicting information is introduced by Phase 2, since there is no change in the state of the light.

Table 6.2 presents the test results for this scenario.

Table 6.2: Scenario 1 Test Results

	Random Forest	Neural Network	Association Rules	Ensemble
	k-Fold Cross-Validation			
Phase 1 w/ Repeated	100%	100%	100%	100%
Phase 1&2 w/ Repeated	95.43%	100%	82.86%	97.33%
Phase 1 w/o Repeated	-	-	-	-
Phase 1&2 w/o Repeated	36%	86%	61%	61%
	Scenario 1 - 20 Tests, discretized into 5			
Phase 1 w/ Repeated	60-80%	80%	100%	100%
Phase 1&2 w/ Repeated	100%	100%	60%	100%
Phase 1 w/o Repeated	40-80%	80%	100%	80%
Phase 1&2 w/o Repeated	80%	80%	100%	80%

All algorithms presented perfect results in the k-fold cross-validation using Phases 1 and good results when introducing Phase 2. There aren't enough entries after removing the duplicates from Phase 1 alone to run K-Fold.

Using k-fold, the lower result when using RF with P12 might be due to conflicting information introduced by Phase 2. Without repetition to indicate the relative weight of the data, poor performance in RF and AR is obtained. The better results with NN, and AR when compared to RF, might be due to the post processing associated with NN and AR. Whereas RF directly returns the prediction result, NN and AR can still return the previous output value if the prediction is between the threshold in NN and no matching rule is found in AR.

It's worth nothing that in AR, the data from Phase 2 will be in higher number than from Phase 1. This means the data that creates the rules pertaining to changing the state of the light will be

filtered out as noise, and the system will not be able to turn the light on or off with this training regime, confirming Phase 2 should not be used with AR.

When running the scenario 1 tests present in annex A.1, all algorithms present at least one combination that results in 100% accuracy, even though there is missing and conflicting information.

Due to the inherent randomness in RF, the accuracy can change from training to training. The best result for both RF and NN comes from using P12, underlining the importance of repetition lending weight to the data. In RF and NN, with Phase 1, both with and without repeated entries, they fail the one test related to the information gap. With P12nR both RF and NN fail the test related to turning off the light due to the conflicting information introduced by Phase 2.

With NN, there is at least one combination that reaches 100% accuracy. The remaining combinations fail the test regarding the missing information. This happens because the end result of the NN is interpolation of the features. With no examples that allows it to get a prediction for the missing information, it will give a value close to 1, and above the upper threshold. Merely increasing the threshold won't be enough, since other predictions for which there is data will be affected.

Below are the NN tests with P12, in order to better understand the NN results.

```
{input:{light_1_p: 0, lux_1: 0, presence_1: 1}, expected: 1, result: 0.8146}
{input:{light_1_p: 1, lux_1: 0, presence_1: 0}, expected: 0, result: 0.0246}
{input:{light_1_p: 0, lux_1: 1, presence_1: 1}, expected: 0, result: 0.7242}
{input:{light_1_p: 0, lux_1: 0, presence_1: 0}, expected: 0, result: 0.0477}
{input:{light_1_p: 1, lux_1: 0, presence_1: 1}, expected: 1, result: 0.7586}
```

The algorithm presents results close to 0 when the expected result is 0 except in test 3, for which there is no information. Since only one feature changes when compared to test 1, the result will be closer to 1 instead.

Association Rules present the best results with just Phase 1, regardless of repetition. This confirms that Phase 2 is not necessary, since Phase 1 has all the situations where the output device changes state. However, without repetition, noise will not be filtered out and will be considered rules. The 100% result with P12nR is due to the number of entries and the order the rules are created in. With a support of 0.1 and with no repeated data, where would need to be more than 10 unique situations for no rules to be created. As it stands, each situation creates its own rule, but since Phase 1 rules are created and found first in prediction, the prediction rate is 100%. Phase 2 information should still not be used. With P12nR, the phase 1 information will be filtered out, making the system fail the tests related to changing the state of the light.

6.5 Scenario 2

In this scenario, there are 3 lamps in a living room. One above the fireplace to give ambient light, one near the sofa for when the user wants to read and the ceiling light, for when more light is

needed. The objective is to control the fireplace light by using a luminosity sensor and the other lights. The target behavior is as follows:

- If there is enough ambient light, the fireplace light remains off.
- The fireplace light is turned on when it's dark and a presence is detected in the room.
- If one of the other lights is turned on, the fireplace light should turn off, and is then turned on when the other light is turned off.
- If the person leaves the room and it's dark, the fireplace light should be turned off.
- Remaining combinations do not alter the state of the fireplace light.

Data creation for this scenario is as follows:

- At 17h00, the light is turned on and there is a 50% chance the person leaves the room between minutes 10 and 50 of the hour, for a random duration between 1 and 5 minutes.
- The interval between the first and second event is of random duration between 5 and 15 seconds.
- The pattern repeats at 18h00 and 21h00, while the light remains on.
- At 19h30, the person leaves the room and returns at 19h59, as to avoid overlapping timestamps.
- At 20h00, the fireplace light is turned off and the ceiling light is turned on, in any order, for a random interval between 15 and 45 minutes. The ceiling light is then turned off, and the fireplace light is turned on.
- The same pattern repeats at 22h00, but with the floor light instead of the ceiling light.
- The light is turned off at 23h00, and the person leaves the room.
- This patterns remains the same on weekends.

For this test, the user sets the discretization interval of the luminosity measure to 5, effectively dividing the range of values into bright (5 to 9) and dark (0 to 4), and a time window of 15 seconds. In this scenario the user is never present in the room with a luminosity of 5 and above, creating a gap in the information. This scenario has the same kind of conflicting information as scenario 1.

Table 6.3 presents the test results for this scenario.

The k-fold tests present largely similar results to scenario 1, with 100% accuracy with just Phase 1 and poor results in RF and AR with both phases and no repeated data.

With RF, it's P1nR instead of P12 that can get the highest accuracy when compared to scenario 1. This might be due to the larger number of features in Scenario 2 affecting how the RF algorithm builds the trees, but there is still one combination that can reach 100% accuracy.

Table 6.3: Scenario 2 Test Results

	Random Forest	Neural Network	Association Rules	Ensemble
	KFold Cross Validation			
Phase 1 w/ Repeated	100%	100%	100%	100%
Phase 1&2 w/ Repeated	96.23%	95.93%	82.27%	96.32%
Phase 1 w/o Repeated	-	-	-	-
Phase 1&2 w/o Repeated	41%	92%	41%	48%
	Scenario 2 - 25 Tests, discretized into 5			
Phase 1 w/ Repeated	80%	80%	100%	100%
Phase 1&2 w/ Repeated	80%	80%	20%	80%
Phase 1 w/o Repeated	80-100%	80%	100%	100%
Phase 1&2 w/o Repeated	80%	60-80%	100%	80%

With NN, no data combination reaches 100% accuracy. The variation in the accuracy of P12nR is due to how close to the thresholds the predictions are from training to training. The higher number of features affects the result so that it fails the same test regarding missing information in all information combinations. Below are the results for the NN tests with P12.

```
{input:
  {light_fire_p: 0, lux_2: 0, presence_2: 1, switch_ceil: 0, switch_floor: 0},
 expected: 1, result: 0.9969}
{input:
  {light_fire_p: 1, lux_2: 0, presence_2: 1, switch_ceil: 1, switch_floor: 0},
 expected: 0, result: 0.0031}
{input:
  {light_fire_p: 1, lux_2: 0, presence_2: 1, switch_ceil: 0, switch_floor: 1},
 expected: 0, result: 0.0031}
{input:
  {light_fire_p: 1, lux_2: 0, presence_2: 0, switch_ceil: 0, switch_floor: 0},
 expected: 0, result: 0.1253}
{input:
  {light_fire_p: 0, lux_2: 1, presence_2: 1, switch_ceil: 0, switch_floor: 0},
 expected: 0, result: 0.9757}
```

In this scenario, when the expected result is 1, the prediction is much closer than in scenario 1. However, it is also much closer in test 5, for which there is no information. The gap between these two predictions is so small that the upper threshold needs to be adjusted to 0.99 in order to have 100% accuracy, but the prediction values might change slightly from training to training, so it is not a sure solution. Test 4 is the situation for which there is conflicting information, so the results isn't as close to 0 as in tests 2 and 3, but it's still below the lower threshold.

For AR, the same observations from scenario 1 hold true, results are the best with Phase 1 information while introducing Phase 2 degrades accuracy, and P12nR work because of the order the rules are created and searched for prediction.

6.6 Scenario 1 with Activity Labels

One of the objectives of this project is to be able to deal with multiple types of values, and not just numerical, as is the case with scenarios 1 and 2.

This scenario is similar to scenario 1, but each sensor outputs labels instead of numerical values. The target behavior is as follows:

- If light is off, the luminosity label is "dark" and the presence label is "reading" or "sewing", the light is turned on.
- If light is off, the luminosity label is "dark" and the presence label is "away" or "pet", the light remains off.
- In the remaining cases, the state of the light remains the same.

Daily data creation rules for this scenario are as follows:

- When a person sits down, they stay for a random duration between 15 and 45 minutes.
- There is a 50% change a person gets up during this interval, for a random duration between 1 and 5 minutes.
- A person sitting down can happen before or after turning on the light.
- The interval between the first and second event is of random duration between 5 and 15 seconds.
- This routine happens 3 times a day on weekdays and 4 time a day on weekends.

Table 6.4 presents the test results for this scenario. Scenario 3 tests use RF and AR, since the NN library was made to use numerical values only and Ensemble depends on the 3 algorithms.

The RF library being used has some problems creating trees with strings as feature values, so some information combinations could not be run. In those that could, the results are congruent with those of scenarios 1 and 2. In P12nR, it fails the same types of tests as in previous scenarios, one related to turning off the light due to conflicting information of Phase 2 and one where information was missing.

Association Rules also present similar results to the previous scenarios. The biggest difference is in P12nR. There are 11 unique feature combinations that create 11 different rules. With a support of 0.1, every rule will be culled as noise, making it so that it fails the same tests it does in P12, tests related to turning the light on or off.

These tests show that at least one algorithm can have 100% accuracy even when missing information and using strings as feature values.

Table 6.4: Scenario 1 w/Labels Test Results

	Random Forest	Association Rules
	KFold Cross Validation	
Phase 1 w/ Repeated	100%	97.93%
Phase 1&2 w/ Repeated	-	86.66%
Phase 1 w/o Repeated	-	-
Phase 1&2 w/o Repeated	-	51%
	Scenario 3 - 8 Tests	
Phase 1 w/ Repeated	-	100%
Phase 1&2 w/ Repeated	-	62.5%
Phase 1 w/o Repeated	-	100%
Phase 1&2 w/o Repeated	75%	62.5%

6.7 Reinforcement

This section will present the tests of the reinforcement mechanic described in section 4.7. This feature was tested using the same data as with scenarios 1 and 2 presented thus far.

6.7.1 Multi-Armed Bandit with UCB

There will be two rounds of tests, executed as such:

1. The algorithm makes a prediction
2. That prediction will be rewarded with 1 point
3. The UCB formula will select a decision
4. That decision will be compared to the expected result
   (a) If the decision is correct, it will be rewarded with 3 points, simulating user feedback
   (b) If the decision is incorrect, it will be rewarded with 0 points, simulating user feedback

Round one will initialize the decisions and round two will take the user feedback from the previous round into account. Table 6.5 presents the test results when using reinforcement with scenario 1, and table 6.6 the results of scenario 2.

In both scenarios, Round 1 shows bad results. This is due to the need to initialize the possible outcomes of the decisions, so the UCB selection formula can be properly applied. The more possible results each decision has, the more rounds will be necessary.

In Round 1, when the algorithms give the correct prediction, the incorrect one will be unrewarded and unexplored and as such returned to the user, which will reward it with zero points. The opposite happens with the incorrect predictions. In that case the correct one will be returned,

Table 6.5: Scenario 1 w/UCB Reinforcement Test Results

	Random Forest	Neural Network	Association Rules	Ensemble
		Round 1		
Phase 1 w/ Repeated	20%	20%	0%	0%
Phase 1&2 w/ Repeated	0%	0%	40%	0%
Phase 1 w/o Repeated	20%	20%	0%	20%
Phase 1&2 w/o Repeated	20%	20%	0%	20%
		Round 2		
Phase 1 w/ Repeated	100%	100%	100%	100%
Phase 1&2 w/ Repeated	100%	100%	100%	100%
Phase 1 w/o Repeated	100%	100%	100%	100%
Phase 1&2 w/o Repeated	100%	100%	100%	100%

Table 6.6: Scenario 2 w/UCB Reinforcement Test Results

	Random Forest	Neural Network	Association Rules	Ensemble
		Round 1		
Phase 1 w/ Repeated	0%	20%	0%	0%
Phase 1&2 w/ Repeated	20%	20%	80%	20%
Phase 1 w/o Repeated	20%	20%	0%	0%
Phase 1&2 w/o Repeated	40%	20%	0%	20%
		Round 2		
Phase 1 w/ Repeated	100%	100%	100%	100%
Phase 1&2 w/ Repeated	100%	100%	100%	100%
Phase 1 w/o Repeated	100%	100%	100%	100%
Phase 1&2 w/o Repeated	100%	100%	100%	100%

and rewarded by the user with 3 points. This results in the complementary percentages of tables 6.2 and 6.3.

In Round 2, all algorithms achieve 100% accuracy in all phase combinations due to the UCB selection formula. The reinforcement mechanic used assures that every information combination on every algorithm reaches 100% accuracy.

6.7.2 Via User Action

This scenario is the same as scenario 2, but where every output device decision in scenario 2 was made by the user, they are now made by the system, which has a 50% chance to return the incorrect result.

Testing is done in two rounds. The first round is without any user corrections. In the second round, every time the system return the incorrect result, the user corrects that action within 5 seconds, as exemplified in section 4.7.2.

Table 6.7: Modified Scenario 2 w/Reinforcement Via User Test Results

	Random Forest	Neural Network	Association Rules	Ensemble
		Before Corrections		
Phase 1 w/ Repeated	40%	20%	40%	40%
Phase 1&2 w/ Repeated	20%	20%	20%	20%
Phase 1 w/o Repeated	60%	20%	20%	20%
Phase 1&2 w/o Repeated	60-80%	20%	20%	20%
		After Corrections		
Phase 1 w/ Repeated	100%	80%	100%	100%
Phase 1&2 w/ Repeated	80%	80%	40%	80%
Phase 1 w/o Repeated	80-100%	80%	100%	80-100%
Phase 1&2 w/o Repeated	60-80%	60%	100%	60-80%

The results show better accuracy across almost every information combination, confirming that the process described in section 4.7.2 has the potential to improve the accuracy of the system.

6.8 Conclusion

This chapter presented the validation of the system by using k-fold cross-validation and tests with two scenarios, along with four training data combinations. The first scenario presents an additional variation with higher level data as described in section 4.8. The MAB with UCB formula was tested with both scenarios and reinforcement via user action was tested with a modified version of scenario 2.

Overall, good results were achieved, which was to be expected considering that the algorithms are being tasked with making predictions with data with which they already trained with, considering that within the context of the project such repetition is to be expected due to the user repeating his patterns.

The best overall method were the Association Rules, which present the highest accuracy with the least amount of data needed. The tests also show that the two reinforcement methods presented can increase the accuracy of the system.

Chapter 7

Conclusions and Future Work

Chapter 2 presented a state of the art on IoT, Smart Home devices and applications and machine learning techniques. Chapter 3 described the objectives of the project. Chapter 4 presented the design decisions of the various features of the system, and chapter 5 the main implementation details.

In chapter 6, validation methods and test are presented and discussed. It is possible to conclude, from the test results, that there is always at least one combination of algorithm and available information that can hit 100% accuracy, that the reinforcement techniques being used, multi-armed bandit with the upper confidence bounds formula and the adjustments to the training data via direct user action, increase the accuracy of the system and that the system can deal with both numerical and string values.

7.1 Main Contributions

The main contributions of this project, as delineated in chapter 3.1 are:

1. Use machine learning to learn user patterns - The system is able to learn user patterns by using machine learning algorithms, the use of which is outlined in section 4.9;

2. Automate devices using patterns learned - The system is able to take control over devices by using the models created, as described in section 4.5;

3. Allow the user to participate in the learning process - With a reinforcement mechanic, the user can mark system decisions as correct or incorrect. Additionally the user's own interactions with the devices are also taken into account. These processes are described in section 4.7.

4. Deal with devices heterogeneously - The project employs a logic that will treat devices in an homogeneous fashion, as described by the creation of the training data, seen in sections 4.4 and 4.7.

Test results in chapter 6 allow for the conclusion that the objectives of the project were accomplished and successfully implemented, with simple scenarios being automated. Tests also allow for the conclusion that the reinforcement methods used, and by extension the participation of the user in the process, add value to the system.

7.2 Future Work

There are still many improvements that can be made to the project. As it stands, each node can only control one device, but it might be useful to control more than one, avoiding the need to have multiple nodes with the same input devices.

More complex scenarios could also be introduced, such as the control of ACs, which the system doesn't support. Even though the current Neural Network library in use supports information other than numeric, adjustments could be made to convert the activity labels into numbers and allow the algorithm to deal with information as described in section 4.8.

There is also the possibility that the person who installs the system is not the same person that uses it. If that were the case, the user might have a lower level of expertise and would not be able to use the system, in which case different approaches in interaction might be necessary, such as voice commands, which allow for the user to more easily interact with the system.

Presently the user can only reinforce one decision at a time, but considering the capacity for nodes to establish their own endpoints, Node-Red could send the information to a web app, where the user could search, filter and reward decisions. This app could also be extended, allowing the user to create rules and send them to Node-Red, as well as have an automatic analysis option to create them so that they can be enabled or corrected, allowing for greater understanding of the underlying behavior of the system.

7.3 Epilogue

In order to make use of the system, the user is asked to install Nodejs, Node-Red and InfluxDB, as well as connect multiple devices to Node-Red and make sure each one sends the correct values, in a DIY solution. When a user already has this kind of expertise, does he benefit from a system that can automate the learning of simple scenarios, where a simple node with user defined rules could maybe achieve the same result? In order to justify the expertise required from the user, systems such as this should be able to handle more complex scenarios. On the other hand, what realistic scenarios might exist in a home where the user cannot create rules that describe it and would prefer to use this kind of system instead? Even assuming such scenarios exist, would someone feel comfortable letting a machine control these complex scenarios in their own home?

There is another consideration when taking into account rules vs. patterns. Rules describe the perfect expected behavior of a device, but user patterns might not. A user might not have the care to optimize his own behavior and always turn off the lights when he should, for example, so in some cases using patterns over rules might not be the best option.

References

[ACWR16] Bikash Agarwal, Antorweep Chakravorty, Tomasz Wiktorski, and Chunming Rong. Enrichment of machine learning based activity classification in smart homes using ensemble learning. In *Proceedings of the 9th International Conference on Utility and Cloud Computing - UCC '16*, pages 196–201, New York, New York, USA, 2016. ACM Press.

[AIS93] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2):207–216, jun 1993.

[App17a] Apple. HomeKit - All Accessories - Apple. https://www.apple.com/us/shop/accessories/all-accessories/homekit, 2017. [Online; accessed 12-June-2017].

[App17b] Apple. Homekit - apple developer. https://developer.apple.com/homekit/, 2017. [Online; accessed 12-June-2017].

[apr] Apriori Nodejs Module, howpublished = https://www.npmjs.com/package/apriori, note = Accessed: 2017-12-06.

[BeO17] BeOnhome. Smart Security Lighting. https://www.beonhome.com/, 2017. [Online; accessed 12-June-2017].

[BK16] Sanjeevani Bhardwaj and Alok Kole. Review and study of internet of things: It's the future. *2016 International Conference on Intelligent Control Power and Instrumentation (ICICPI)*, pages 47–50, 2016.

[BL14] Michael Blackstock and Rodger Lea. Toward a Distributed Data Flow Platform for the Web of Things (Distributed Node-RED). In *Proceedings of the 5th International Workshop on Web of Things - WoT '14*, pages 34–39, New York, New York, USA, 2014. ACM Press.

[CCTK13] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. CASAS: A smart home in a box. *Computer*, 46(7):62–69, 2013.

[CMV+10] Robert Cloutier, Gerrit Muller, Dinesh Verma, Roshanak Nilchiani, Eirik Hole, and Mary Bone. The concept of reference architectures. *Syst. Eng.*, 13(1):14–27, February 2010.

[DF14] Ted Dunning and Ellen Friedman. *Time Series Databases New Ways to Store and Access Data*. 2014.

[Die95] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM Computing Surveys*, 27(3):326–327, 1995.

[DKS95] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and Unsupervised Discretization of Continuous Features. *Machine Learning Proceedings 1995*, 0:194–202, 1995.

[EFGK03] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[FFK+13] Steven K. Firth, Farid Fouchal, Tom Kane, Vanda Dimitriou, and Tarek M. Hassan. Decision support systems for domestic retrofit provision using smart home data streams. *CIB W78 2013 The 30th International Conference on Applications of IT in the AEC Industry. Move Towards Smart Buildings: Infrastructures and Cities*, 2013.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning–book. *MIT Press*, 521(7553):800, 2016.

[GE11] Raja Giryes and Michael Elad. Reinforcement Learning: A Survey. *Eur. Signal Process. Conf.*, pages 1475 – 1479, 2011.

[Gia15] Developing IoT applications in the Fog: A Distributed Dataflow approach. In *2015 5th International Conference on the Internet of Things (IOT)*, pages 155–162. IEEE, oct 2015.

[HHKT16] Li-Yu Hu, Min-Wei Huang, Shih-Wen Ke, and Chih-Fong Tsai. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5(1):1304, 2016.

[IFT] IFTTT. Discover IFTTT and Applets.

[Inf16a] InfluxData. Benchmarking InfluxDB vs. MongoDB for Time-Series Data, Metrics & Management. 2016.

[Inf16b] InfluxData. Benchmarking InfluxDB vs. OpenTSDB for Time-Series Data, Metrics & Management. 2016.

[JDJ00] A.K. Jain, P.W. Duin, and Jianchang Mao. Statistical pattern recognition: a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4–37, 2000.

[Kee12] Valerie Lampkin. Weng Tat Leong. Leonardo Olivera. Sweta Rawat. Nagesh Subrahmanyam. Rong Xiang. Martin Keen. *Building Smarter Planet Solutions with MQTT and IBM WebSphere MQ Telemetry*. IBM, 2012.

[KGV16] David Burth Kurka, Alan Godoy, and Fernando J. Von Zuben. Birds of a feather tweet together: Computational techniques to understand user communities in social networks. *CEUR Workshop Proceedings*, 1691:21–27, 2016.

[Koh95] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. *Appears in the International Joint Conference on Articial Intelligence (IJCAI)*, 5:1–7, 1995.

[KP14] Volodymyr Kuleshov and Doina Precup. Algorithms for multi-armed bandit problems. *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm - SODA '06*, 1:928–936, feb 2014.

[LB04] Adele Cutler Leo Breiman. Random Forests. https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm/, 2004. [Online; accessed 12-June-2017].

[LCL16] Gabriele Lobaccaro, Salvatore Carlucci, and Erica Löfström. A Review of Systems and Technologies for Smart Homes and Smart Grids. *Energies*, 9(5):348, may 2016.

[lim] Limdu Nodejs Module, howpublished = https://www.npmjs.com/package/limdu, note = Accessed: 2017-12-06.

[LN16] Emil Laftchiev and Daniel Nikovski. An IoT system to estimate personal thermal comfort. In *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*, pages 672–677. IEEE, dec 2016.

[LNB12] Quynh Lê, Hoang Boi Nguyen, and Tony Barnett. Smart Homes for Older People: Positive Aging in a Digital World. *Future Internet*, 4(4):607–617, 2012.

[MF10] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 6462 LNCS:242–259, 2010.

[MMST16] Julien Mineraud, Oleksiy Mazhelis, Xiang Su, and Sasu Tarkoma. A gap analysis of Internet-of-Things platforms. *Computer Communications*, 89-90:5–16, 2016.

[Mona] MongoDB. MongoDB for GIANT Ideas - A Distributed, Scalable Monitoring System.

[Monb] MongoDB. Schema Design in Time Series Data for MongoDB.

[MSDC12] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[NES17a] NEST. Works with NEST Store. https://workswith.nest.com/?devices=thermostat, 2017. [Online; accessed 12-June-2017].

[NES17b] NESTr. Meet the NEST Thermostat. https://nest.com/thermostat/meet-nest-thermostat/, 2017. [Online; accessed 12-June-2017].

[nod] Node-Red InfluxDB Nodes, howpublished = https://www.npmjs.com/package/node-red-contrib-influxdb, note = Accessed: 2017-12-06.

[NR17] Node-Red. Node-Red. https://nodered.org/, 2017. [Online; accessed 12-June-2017].

[Ope] OpenTSDB. OpenTSDB - A Distributed, Scalable Monitoring System.

REFERENCES

[Pan17] Kasey Panetta. Gartner Top Strategic Predictions for 2018 and Beyond. https://www.gartner.com/smarterwithgartner/gartner-top-strategic-predictions-for-2018-and-beyond/, 2017. [Online; accessed 28-February-2018].

[Put90] Martin L. Puterman. Chapter 8 Markov decision processes. *Handbooks in Operations Research and Management Science*, 2:331–434, 1990.

[rf-] Random Forest Classifier Nodejs Module, howpublished = https://www.npmjs.com/package/random-forest-classifier, note = Accessed: 2017-12-13.

[Ris06] I Rish. An Empirical Study of the Naive Bayes Classifier. (January 2001):41–46, 2006.

[Sam] Samsung. SmartThings. Add a little smartness to your things.

[Sam17] Samsung. A Safer and Smarter Home in the Palm of your Hands - SmartThings. https://shop.smartthings.com/, 2017. [Online; accessed 12-June-2017].

[Seg07] Toby Segaran, editor. *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. O'Reilly Media, Oxford, 2007.

[Som] Somfy. Alarms, Security Cameras and Smart Home Solutions.

[Sut92] Richard S. Sutton. Introduction: The challenge of reinforcement learning. *Machine Learning*, 8(3-4):225–227, 1992.

[vdM17] Rob van der Meulen. Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016. http://www.gartner.com/newsroom/id/3598917, 2017. [Online; accessed 12-June-2017].

[Vee16] Fjodor Van Veen. The Neural Network Zoo - The Asimov Institute. http://www.asimovinstitute.org/neural-network-zoo/, 2016. [Online; accessed 28-February-2018].

[WB14] Gerald Wagenknecht and Torsten Ingo Braun. *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, volume 8638. 2014.

[WD92] Christopher J.C.H. Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, 8(3/4):279–292, 1992.

[WHHB15] Charlie Wilson, Tom Hargreaves, and Richard Hauxwell-Baldwin. Smart homes and their users: a systematic analysis and key challenges. *Personal and Ubiquitous Computing*, 19(2):463–476, 2015.

[WIN17] WINK. WINK. https://www.wink.com/about/, 2017. [Online; accessed 12-June-2017].

[WX14] Zhe Wang and Xiangyang Xue. Support Vector Machines Applications. pages 23–49, 2014.

[YN13] R. Yang and M. W. Newman. Learning from a Learning Thermostat : Lessons for Intelligent Systems for the Home. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing (UbiComp 2013)*, pages 93–102, 2013.

Appendix A

Scenario Tests

A.1 Scenario 1 Tests

Table A.1: Tests for Scenario 1

light_1_p	lux_1	presence_1	expected output
0	0	1	1
0	1	1	1
0	2	1	1
0	3	1	1
0	4	1	1
1	0	1	0
1	1	1	0
1	2	1	0
1	3	1	0
1	4	1	0
0	5	1	1
0	6	1	1
0	7	1	1
0	8	1	1
0	9	1	1
0	1	1	0
0	2	1	0
0	3	1	0
0	4	1	0
0	5	1	0

A.2 Scenario 2 Tests

Table A.2: Tests for Scenario 2

light_fire_p	lux_2	presence_2	switch_ceil	switch_floor	expected output
0	0	1	0	0	1
0	1	1	0	0	1
0	2	1	0	0	1
0	3	1	0	0	1
0	4	1	0	0	1
0	5	1	0	0	0
0	6	1	0	0	0
0	7	1	0	0	0
0	8	1	0	0	0
0	9	1	0	0	0
1	0	1	1	0	0
1	1	1	1	0	0
1	2	1	1	0	0
1	3	1	1	0	0
1	4	1	1	0	0
1	0	1	0	1	0
1	1	1	0	1	0
1	2	1	0	1	0
1	3	1	0	1	0
1	4	1	0	1	0
1	0	0	0	0	0
1	1	0	0	0	0
1	2	0	0	0	0
1	3	0	0	0	0
1	4	0	0	0	0

A.3 Scenario 1 with Activity Labels Tests

Table A.3: Scenario 1 with Activity Labels Tests

light_1_p	lux_1	presence_1	expected output
off	dark	reading	on
off	dark	tv	on
on	dark	away	off
off	dark	away	off
off	dark	pet	off
off	bright	reading	off
off	bright	tv	off
off	bright	pet	off